

**ALTERA**®



**SOPC**

**WORLD**

2 0 0 2



**SOPC**  
**WORLD**  
2 0 0 2

## SignalTap II



SignalTap<sup>®</sup> II

ALTERA.

# Agenda

- SignalTap II Interface in Quartus II S/W
- SignalTap II Demonstration
  - Initial Compilation
  - Second Compilation:  
Using SignalTap II Logic Analyzer
- Conclusions

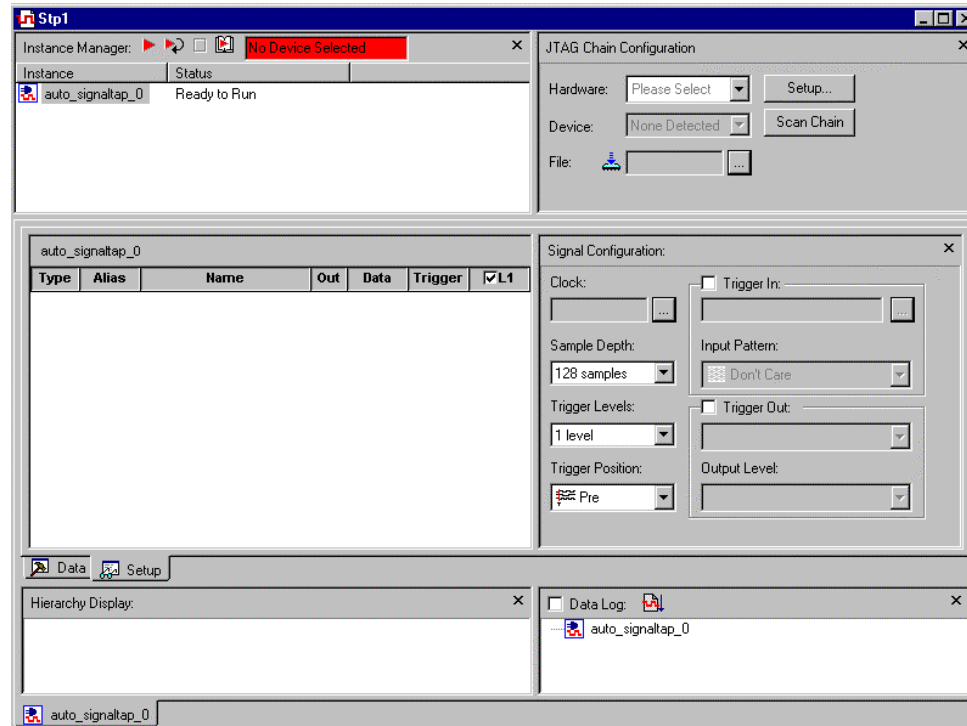


**SOPC**  
**WORLD**  
2 0 0 2

# SignalTap II Interface in Quartus II Software

# SignalTap II File (.stp)

- Creating SignalTap<sup>®</sup> II Logic Analyzer File
  - Choose New (File Menu)
  - Click the Other Files Tab & Select SignalTap II File
  - Click OK



# SignalTap II File

The screenshot shows the SignalTap II software interface with several panels and windows. Red arrows point to the following components:

- Instance Manager:** A window showing a table with one instance named 'auto\_signaltap\_0' with a status of 'Ready to Run'. A red arrow points to this window from the label 'Instance Manager' on the left.
- JTAG Chain Configuration:** A window with fields for Hardware, Device, and File, and buttons for Setup, Scan Chain, and a file browser. A red arrow points to this window from the label 'JTAG Chain Configuratio' on the right.
- Signal Configuration:** A window with settings for Clock, Sample Depth (128 samples), Trigger Levels (1 level), Trigger Position (Pre), and Trigger In/Out options. A red arrow points to this window from the label 'Signal Configuratio' on the right.
- Hierarchy Display:** A window at the bottom left showing a tree view. A red arrow points to this window from the label 'Hierarchy Display' at the bottom.
- Data Log:** A window at the bottom right showing a list of data logs, including 'auto\_signaltap\_0'. A red arrow points to this window from the label 'Data Log' at the bottom.

Hierarchy Display

Data Log

# SignalTap II File: Acquisition Clock

## ■ Acquisition Clock

- For Best Results, Assign Only Global Clock as the SignalTap II Clock Signal
- Without Assigning Clock Signal, Quartus II Software Creates Clock Pin `auto_stp_external_clock`

**Acquisition Clock** →

**Sample Depth** →

**Trigger Levels** →

**Trigger Position** →

← **Trigger In/Out**

# SignalTap II File: Trigger Pattern

## ■ Sample Depth

- Set Number of Samples Stored for Each Input Signal
  - 0 to 128K Sample Depth

## ■ Trigger Levels

- Configure Analyzer with up to 10 Trigger Levels

## ■ Trigger Position

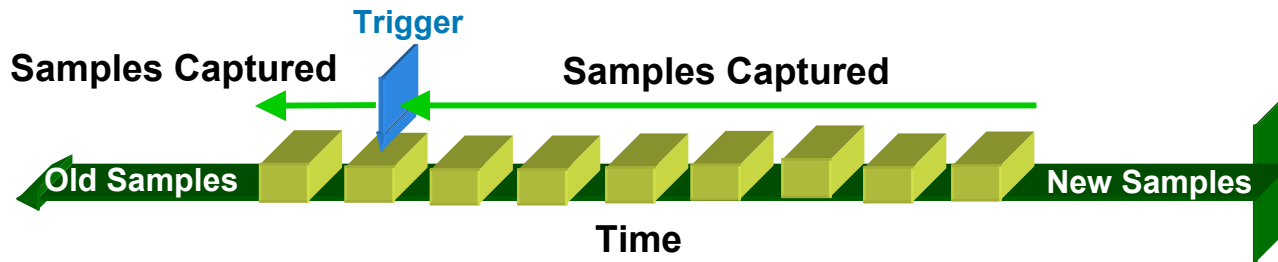
- Specify Amount of Data Captured by SignalTap II Logic Analyzer that Should be Acquired before the Trigger as well as Amount that Should be Acquired after the Trigger



# Trigger Position

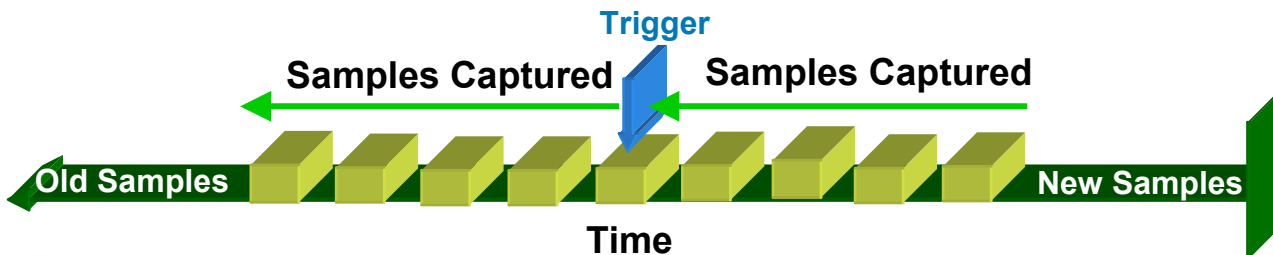
## ■ Pre-Trigger

- Captures Signals that Occur Immediately after Triggering (12% Pre-Trigger, 88% Post-Trigger)



## ■ Center Trigger

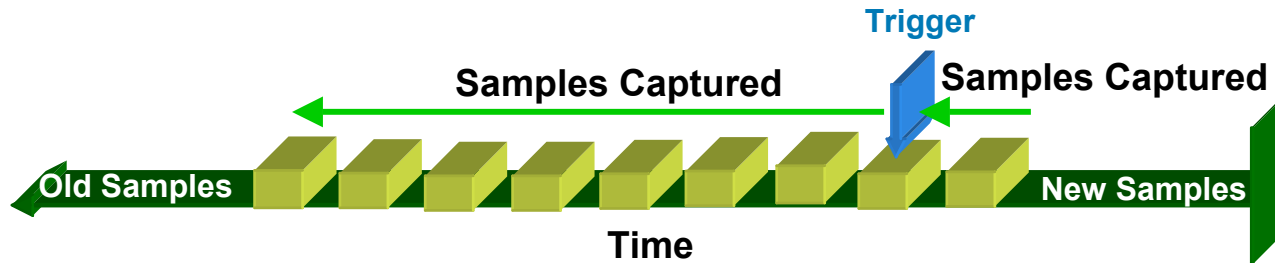
- Captures Signals before & after Triggering (50% Pre-Trigger, 50% Post-Trigger)



# Trigger Position

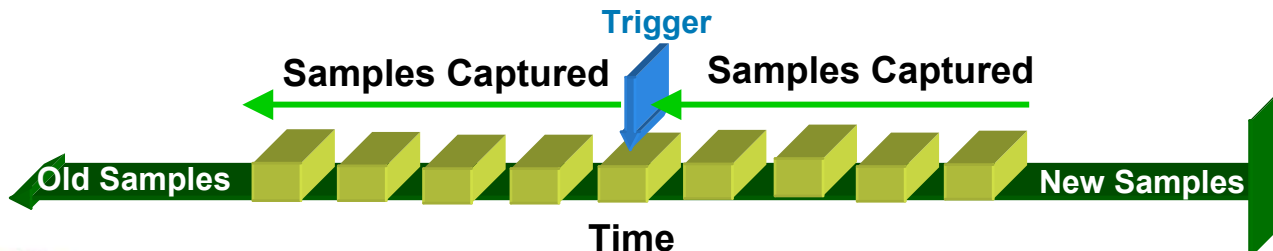
## ■ Post-Trigger

- Captures Signal that Occur Immediately before Triggering (88% Pre-Trigger, 12% Post-Trigger)



## ■ Continuous Trigger

- Captures Signals Indefinitely Until Stopped Manually (Useful When Using the Trigger Out Feature)



# SignalTap II File:Trigger In/Out

## ■ Trigger In












- Any I/O Pin Can Trigger the SignalTap II Analyzer
- Pin `auto_stp_trigger_in_0` Generated in Device
- Trigger Input Can be Set to Recognize High, Low, Rising Edge, Falling Edge, Either Edge, or Don't Care Condition

## ■ Trigger Out

- Spare I/O Pin that Is Set as Trigger Output Signal That Indicates When Trigger Pattern Occurs
- Pin `auto_stp_trigger_out_0` Generated in Device
- Output Pulse Polarity Is Programmable

# SignalTap II File: Debug Ports

- Routing SignalTap II Signal to Spare I/O Pin for Capture by Logic Analyzer
- Quartus II Software Automatically Generates a Pin
  - Debug Port Pin Name Is `stp_debug_out_1_n`
    - n Is a Number Representing the Order in Which the Debug Port Pin Occurs in the Signal List

Type	Alias	Name	Out	Data	Trigger	<input checked="" type="checkbox"/> L1
		CNT_ONE_ENABLE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		CNT_ONE0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	—
		CNT_ONE1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	—
		CNT_ONE2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	—
		CNT_ONE3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	—

Debug Port On

Debug Port Off

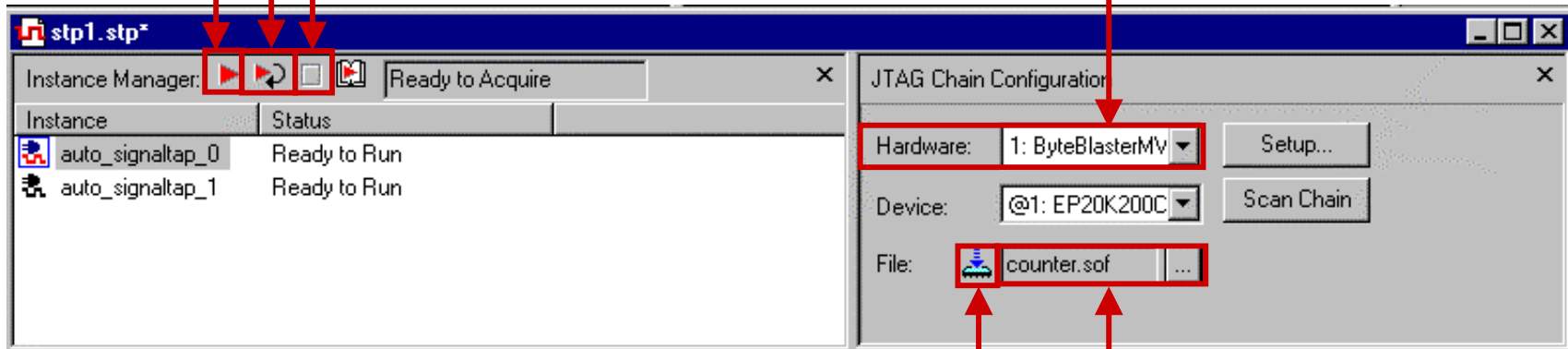
# SignalTap II File: Control Settings

Stop the Logic Analyzer

Run Continuous

Run the Embedded  
Logic Analyzer


Specify Communication  
Hardware

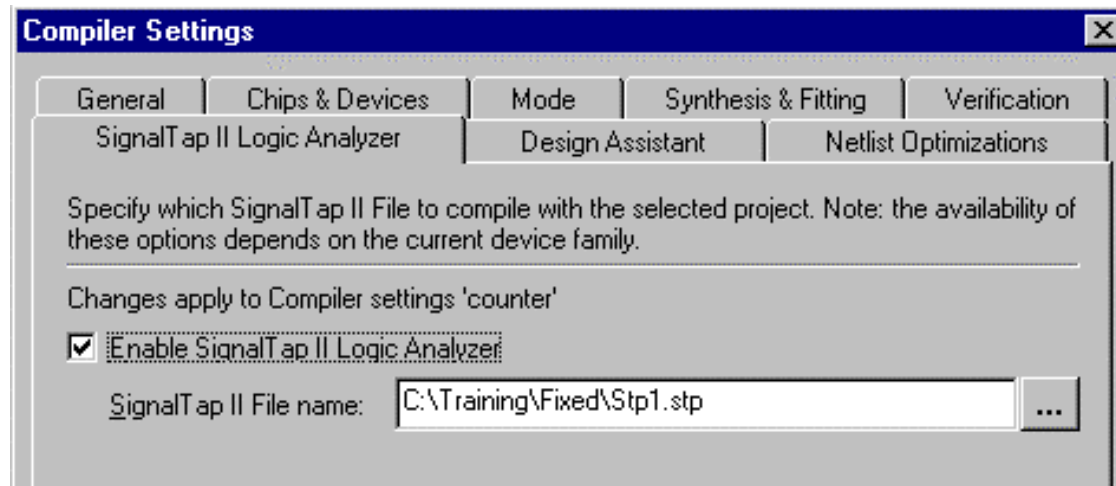


Specify Configuration  
File (.sof or .cdf)

Download SOF file

# Specify SignalTap II File

- Choose Compiler Settings (Processing Menu)
- Click the SignalTap II Logic Analyzer Tab
- Turn on Enable SignalTap II Logic Analyzer
- In the SignalTap II File Name box:
  - Type the Name of the SignalTap II File (.stp) for Compilation
  - Or Select a File Name with Browse Button 



# Project Compilation

- Design Recompile Required When Any of These Parameters Change
  - Acquisition Clock
  - Number of Channels
  - Sample Depth
  - Debug Ports
  - Trigger in/out Ports
- Design Recompile Not Required When
  - Changing Trigger Pattern or Position
  - Modifying Trigger Levels
  - Starting or Stopping SignalTap II Logic Analyzer
- Internal Nodes to Be Analyzed Should be Inputs, Outputs, Register Outputs, or Memory Outputs

# SignalTap II Demonstration

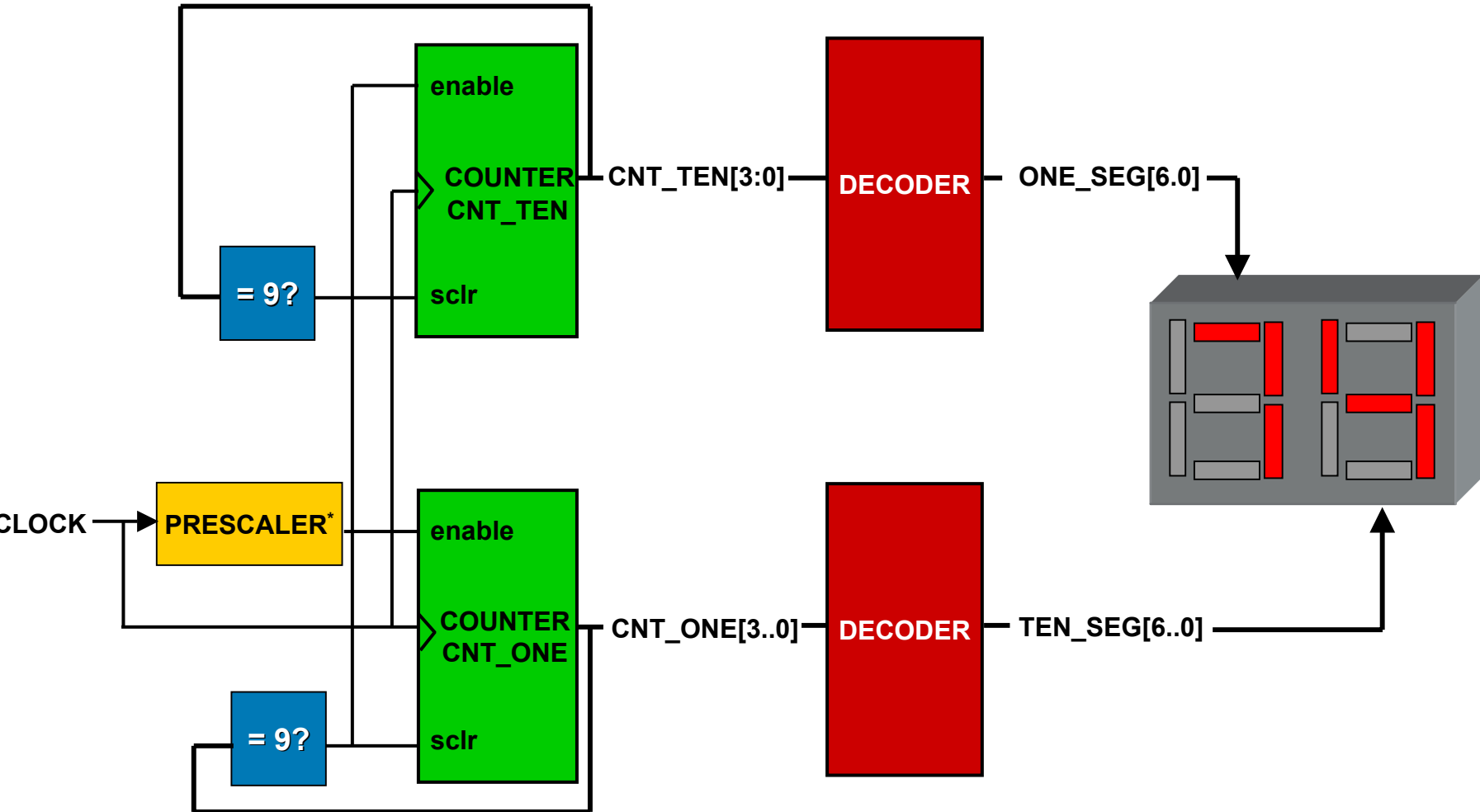
- Description of the Test Case
- Initial Compilation
  - Observing the Malfunction
- Second Compilation
  - Creating the Embedded Logic Analyzer
  - Debugging Design
- Third Compilation
  - Fixing the Design Issue



# Design: Counter from 00 to 99

- Function: Two 4-Bit Counters Cascaded to Drive a Pair of Seven-Segment LEDs that Count from 0 to 99
- Top Level Name: Counter
- Target Device : EP20K200EFC484-2X
- Tools
  - Synthesis: LeonardoSpectrum
  - Fitter: Quartus II Version. 2.1
- Place Source Design Files in **C:\Training\Synthesis** Directory
- Place pins.tcl File in **C:\Training** Directory

# Counter Block Diagram



\* : The Prescaler Generates a Counter Enable with a Frequency of about 1 Hz

# Synthesize Design (1/2)

- Launch LeonardoSpectrum Tool

- Select Mode Quick Setup



- Working Directory: **C:\Training\Synthesis**
- Open Files: **counter.vhd**
- Technology: **APEX 20KE**
- Device: **EP20K200EFC484**
- Speed Grade: **-2X**
- Output File: **C:\Training\Synthesis\counter.edf**

- Click on Run Flow

# Synthesize Design (2/2)

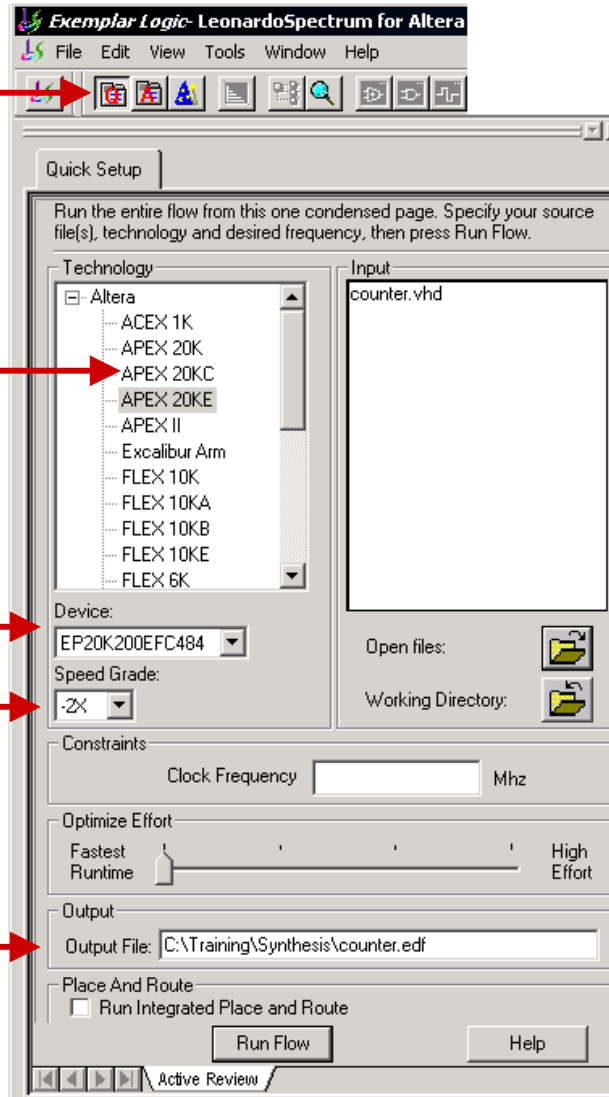
Quick Setup

Technology

Device

Speed Grade

Output File



Source File

Working Directory

# Create Quartus II Project

- Launch the Quartus II Software
- Choose New Project Wizard (File menu) to Create a New Quartus II Project
  - Directory Name: **C:\Training\Fitting**
  - Project Name: **counter**
  - Top Level Name: **counter**
- Click Next

New Project Wizard: Directory, Name, and Top-Level Entity [page 1 of 6]

What is the working directory for this project? This directory will contain design files and other related files associated with this project. If you type a directory name that does not exist, Quartus II can create it for you.

C:\Training\Fitting

What is the name of this project? If you wish, you can use the name of the project's top-level design entity.

counter

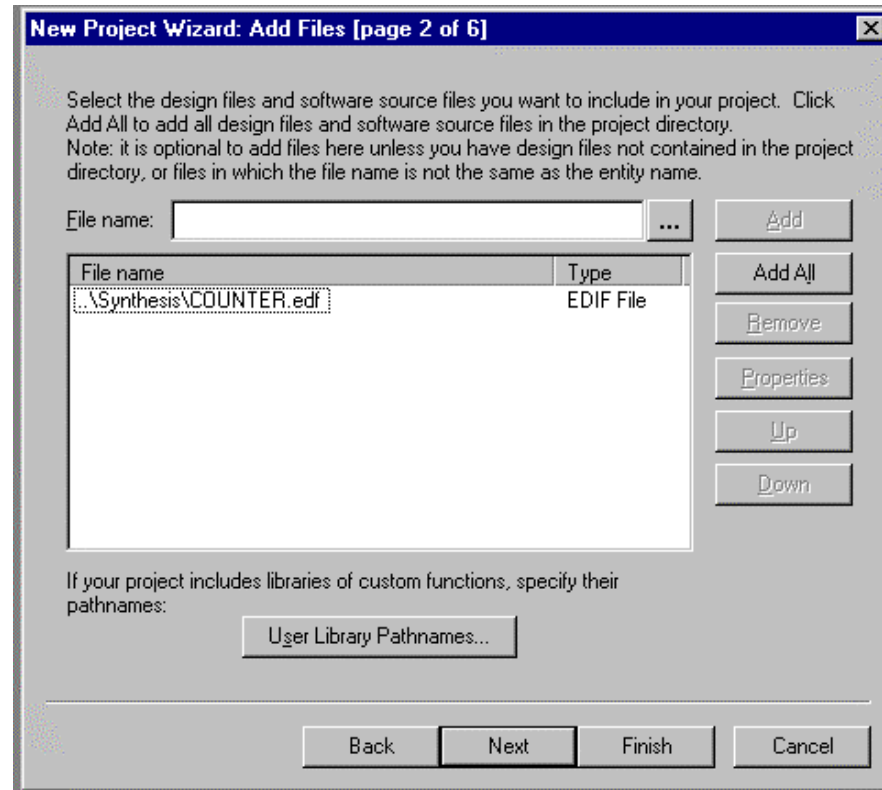
What is the name of the top-level design entity in your project? The Quartus II software will automatically create Compiler and Simulator settings for the top-level entity you specify in this wizard. After you create a project, you can add more top-level entities and create Compiler and Simulator settings for them with commands on the Processing menu.

counter

Back Next Finish Cancel

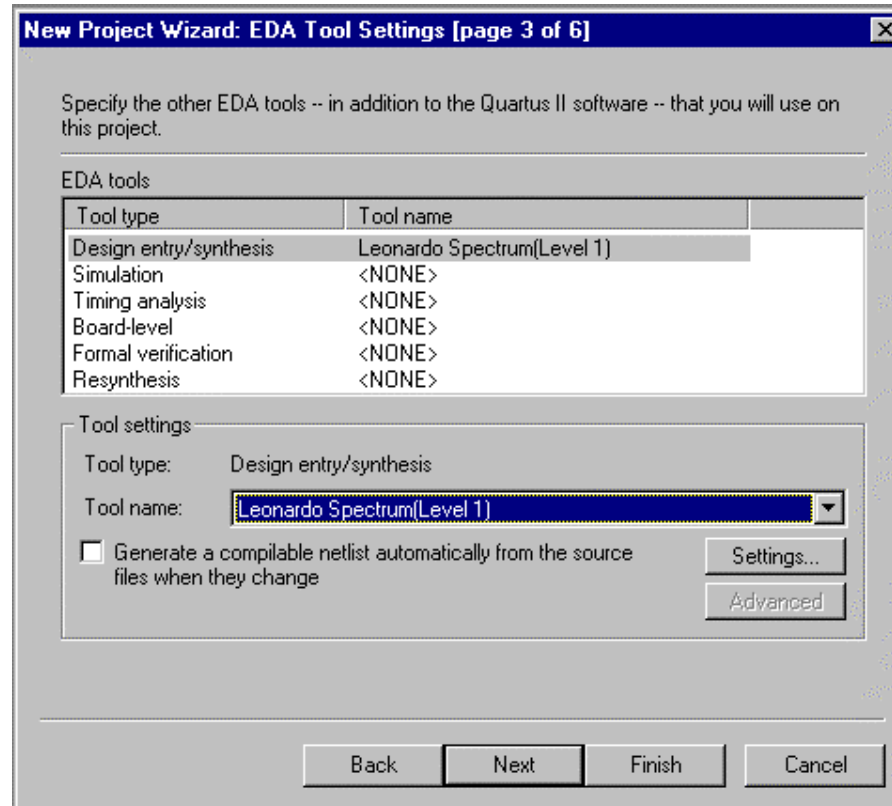
# Create Quartus II Project

- Add Following Files in Project
  - C:\Training\Synthesis\counter.edf
- Click Next
- Click Finish



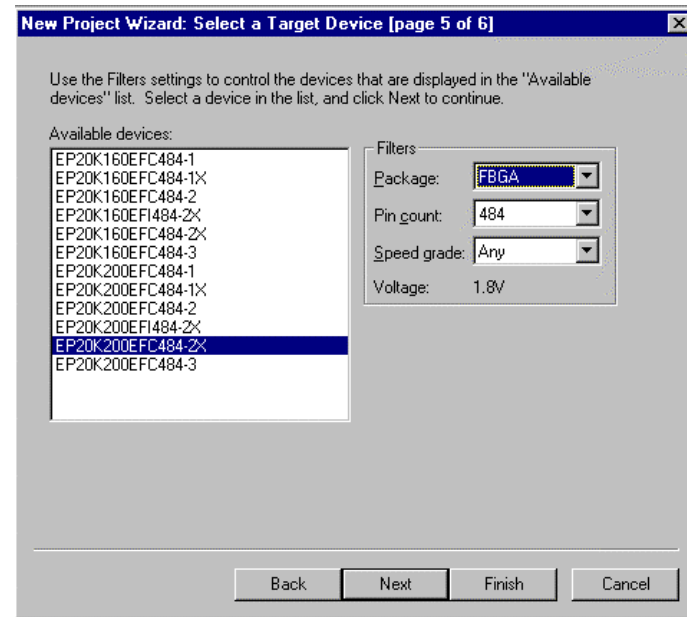
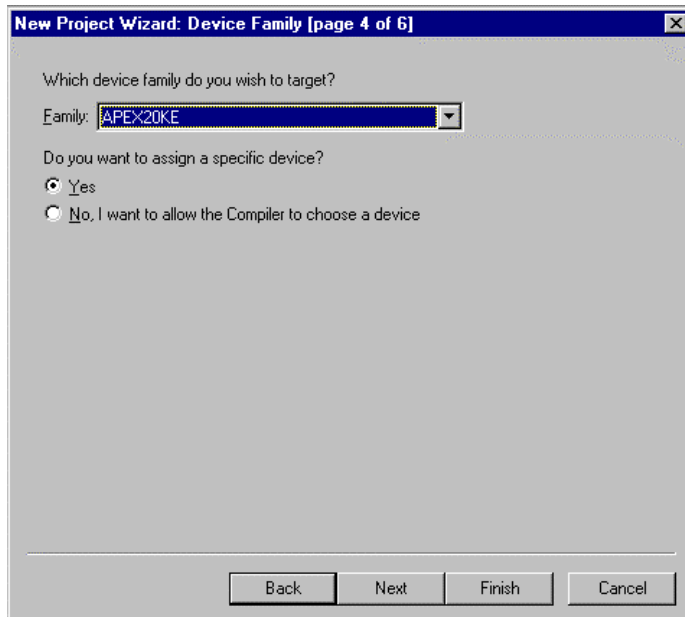
# Create Project: Set EDA Tool

- Select LeonardoSpectrum Tool (Level 1) as Design Entry/Synthesis Tool
- Click Next



# Create Project: Select Device

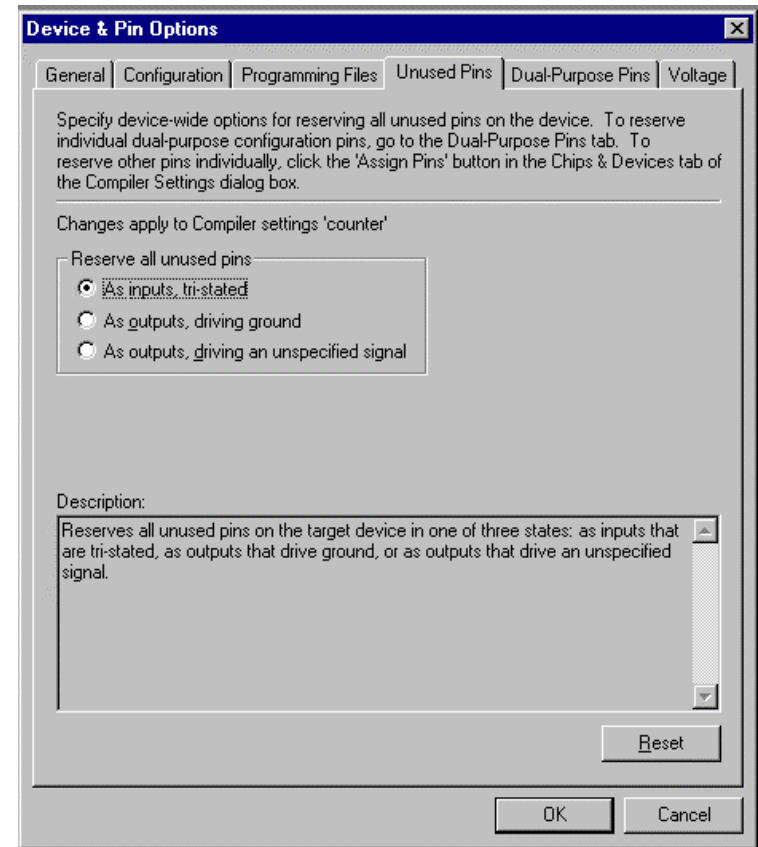
- Select APEX & Yes to Assign Specific Device
- Click Next
- Select EP20K200EFC484-2X
- Click Finish





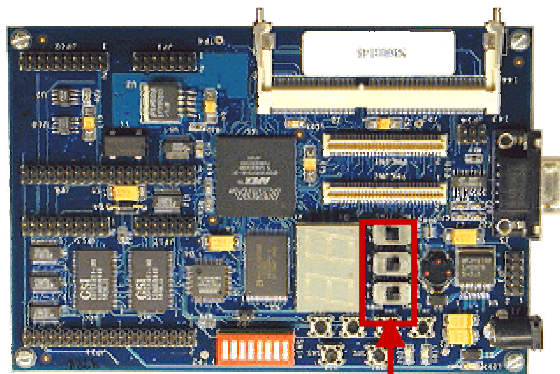
# Set All Unused I/Os as Tri-Stated

- Choose Compiler Settings (Processing Menu)
- Click the Chips & Devices Tab
- Click Device & Pin Options Button
- Click the Unused Pins Tab
  - Select **As inputs, tri-stated**

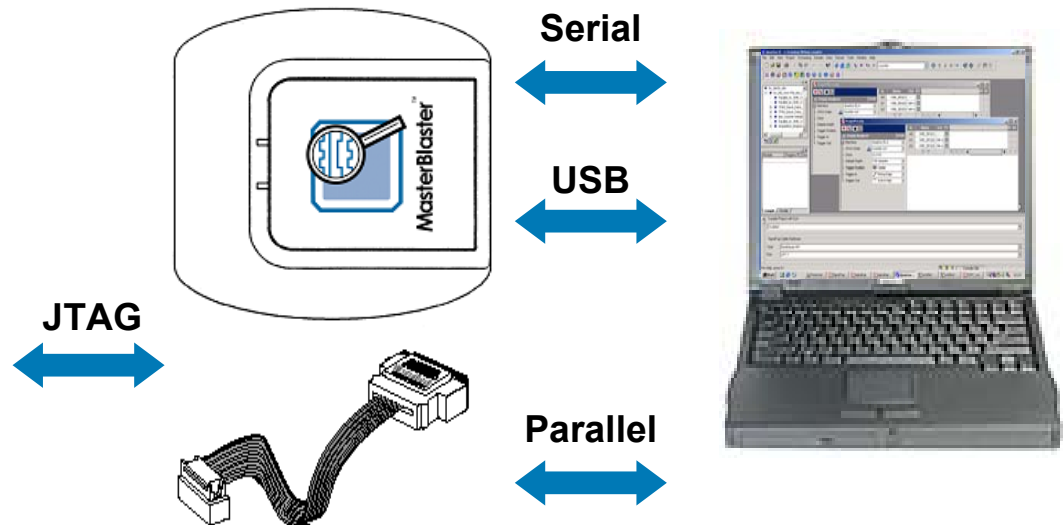


# Hardware Setup

- Nios Demo Board with APEX EP20K200EFC484-2X Device
- DC Power Supply
- Device Download Cable (MasterBlaster or ByteBlasterMV Cable)

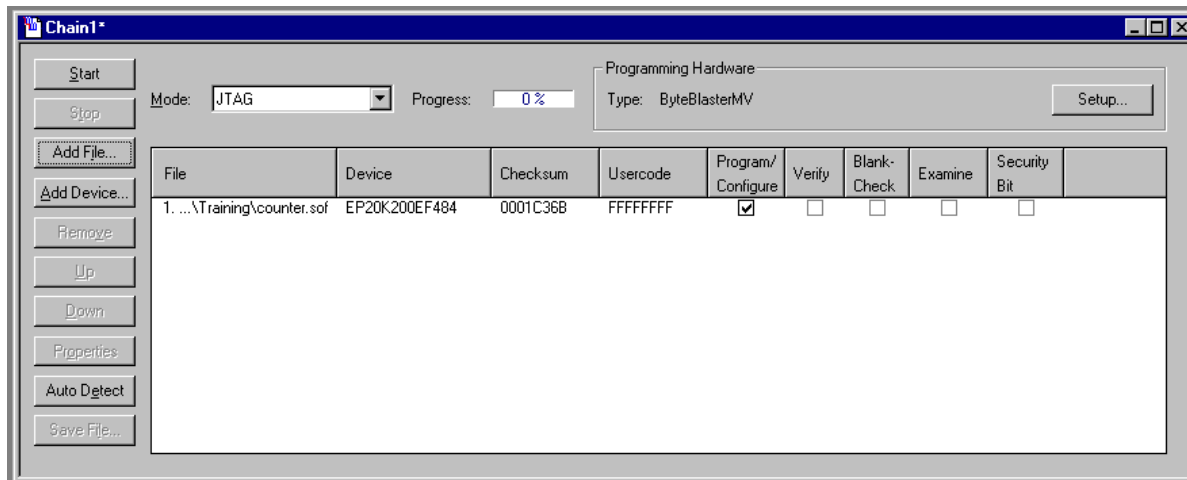


SW10 « JTAG »  
SW9 « JTAG »  
SW8 « APEX »



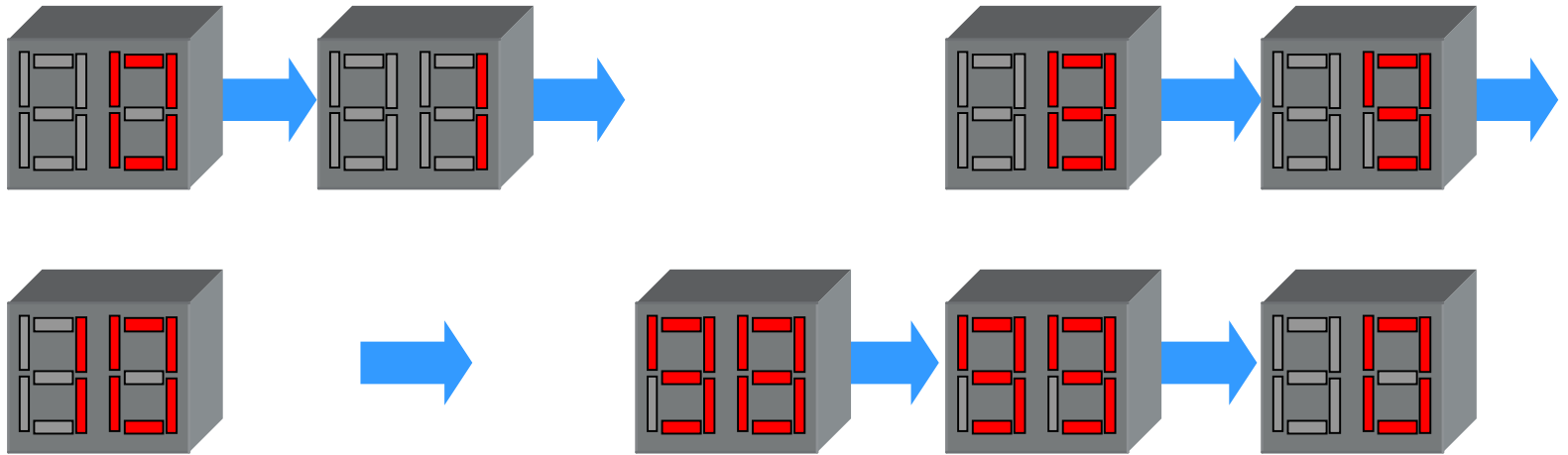
# Compile Project & Configuration

- Choose Start Compilation (Processing menu)
- Choose Open Programmer (Processing menu)
  - Click Add File
    - Select the File counter.sof
  - Programming Hardware : ByteBlasterMV
  - Mode : JTAG
  - Check Option Program/Configure
- Click Start



# Observing the Malfunction

- Number 9 Never Appears on the Least & Most Significant Digit



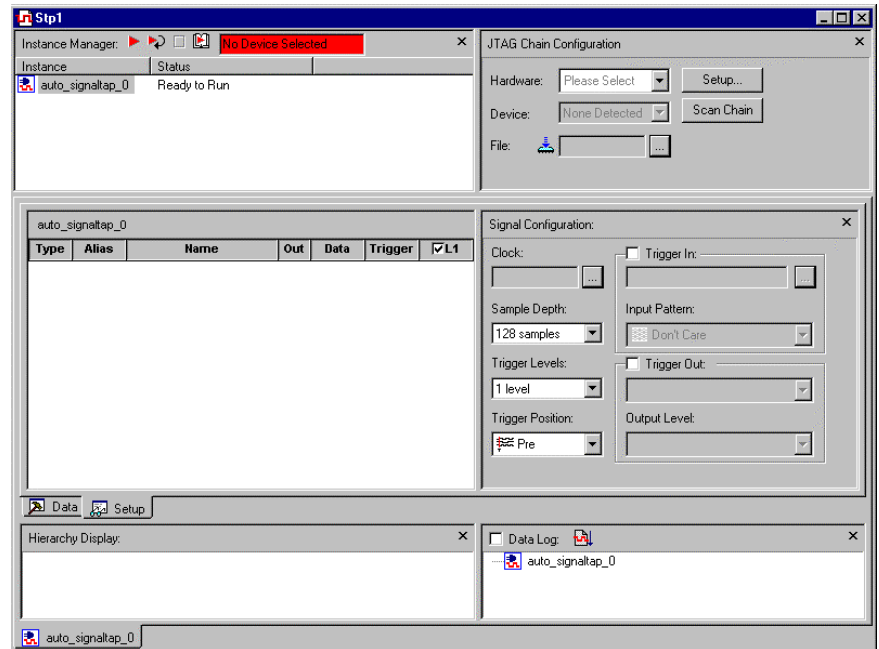
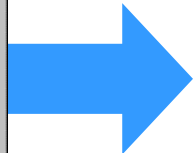
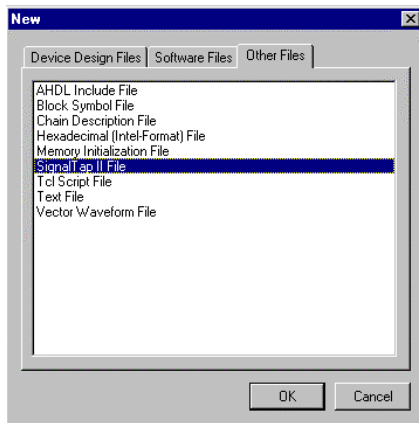
- SignalTap II Analyzer Used to Monitor Control Signals of LEDs

# Essential Steps

- Configure SignalTap II Logic Analyzer
  - Select Nodes for Analysis
  - Select Acquisition Clock
  - Set Sample Depth & Trigger Options
  - Enable the Logic Analyzer & Compile
- Configure Device
- Set the Trigger Pattern
- Run the SignalTap II Logic Analyzer

# Configure Logic Analyzer

- Choose New (File Menu)
- Click the Other Files Tab & Select SignalTap II File
- Click OK

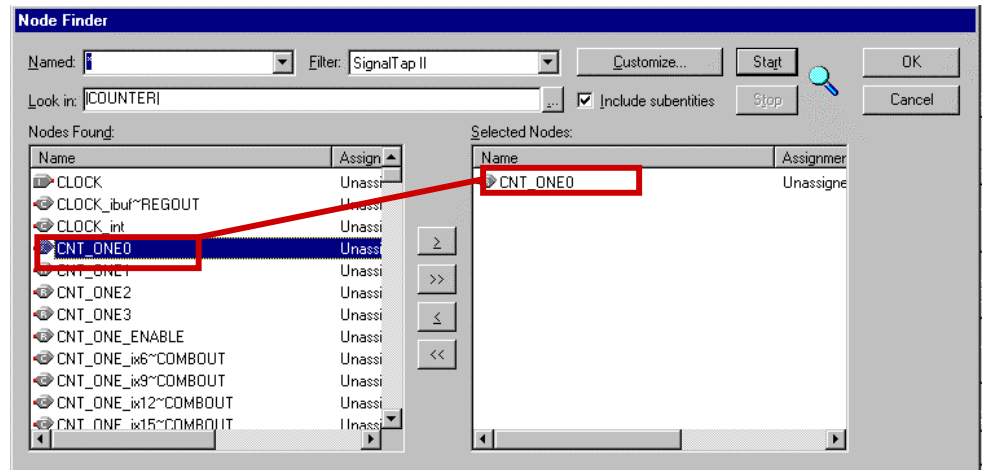


- Or Select 

- Save the SignalTap II File as **analysis.stp**

# Analyzer 1: Select Nodes

- Open Node Finder Window
  - Double Click on the Signal Viewer
- Click Start to List All Pins & Internal Nodes
- Add Signal CNT\_ONE\_0 to Selected Nodes List
- Repeat
  - CNT\_ONE\_1
  - CNT\_ONE\_2
  - CNT\_ONE\_3
  - CNT\_ONE\_ENABLE
- Click OK



# Analyzer 1: Acquisition Clock

- Add Clock Input Pin from Node Finder

**Node Finder**

Named: [ ] Filter: Design Entry (all names) [Customize...] [Start] [OK]

Look in: [ICOUNTER1] [Include subtentities] [Stop] [Cancel]

Name	Assignments	Type
CLOCK	counter	Input
CLOCK_ibuf	Unassigned	Combinatori
CLOCK~out0	Unassigned	Combinatori
CNT_ONE_ENABLE	Unassigned	Registered
CNT_ONE_ix6	Unassigned	Combinatori
CNT_ONE_ix9	Unassigned	Combinatori
CNT_ONE_ix12	Unassigned	Combinatori
CNT_ONE_ix15	Unassigned	Combinatori
CNT_ONE_nx6	Unassigned	Combinatori
CNT_ONE_nx12	Unassigned	Combinatori
CNT_ONE_nx17	Unassigned	Combinatori
CNT ONE 0	Unassigned	Registered

**Selected Nodes:**

Name	Assignments	Type
CLOCK	counter	Input

**Signal Configuration:**

Clock: [ ] [...]

Sample Depth: 128 samples

Trigger Levels: 1 level

Trigger Position: Pre

Trigger In: [ ] [...]

Input Pattern: Don't Care

Trigger Out: [ ] [...]

Output Level: [ ] [...]

Open Node  
Finder for Clock



# Configure Analyzer 1

- Configure Logic Analyzer 1 (auto\_signaltap\_0) as Follows

128 Samples

Single Level

Center

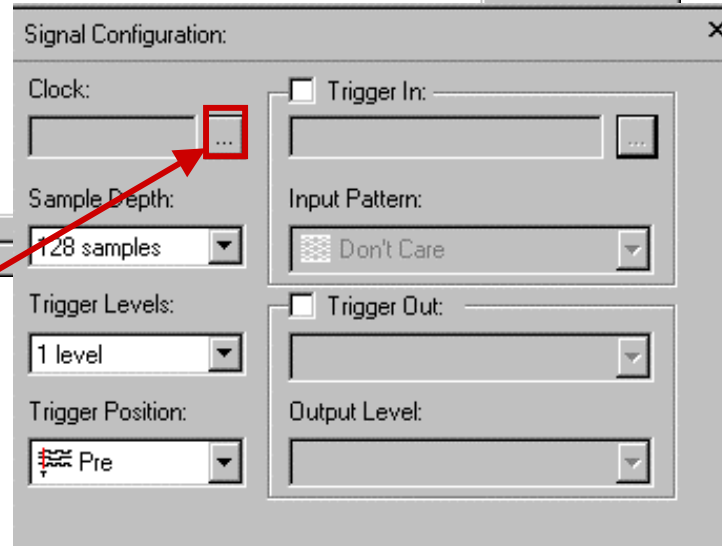
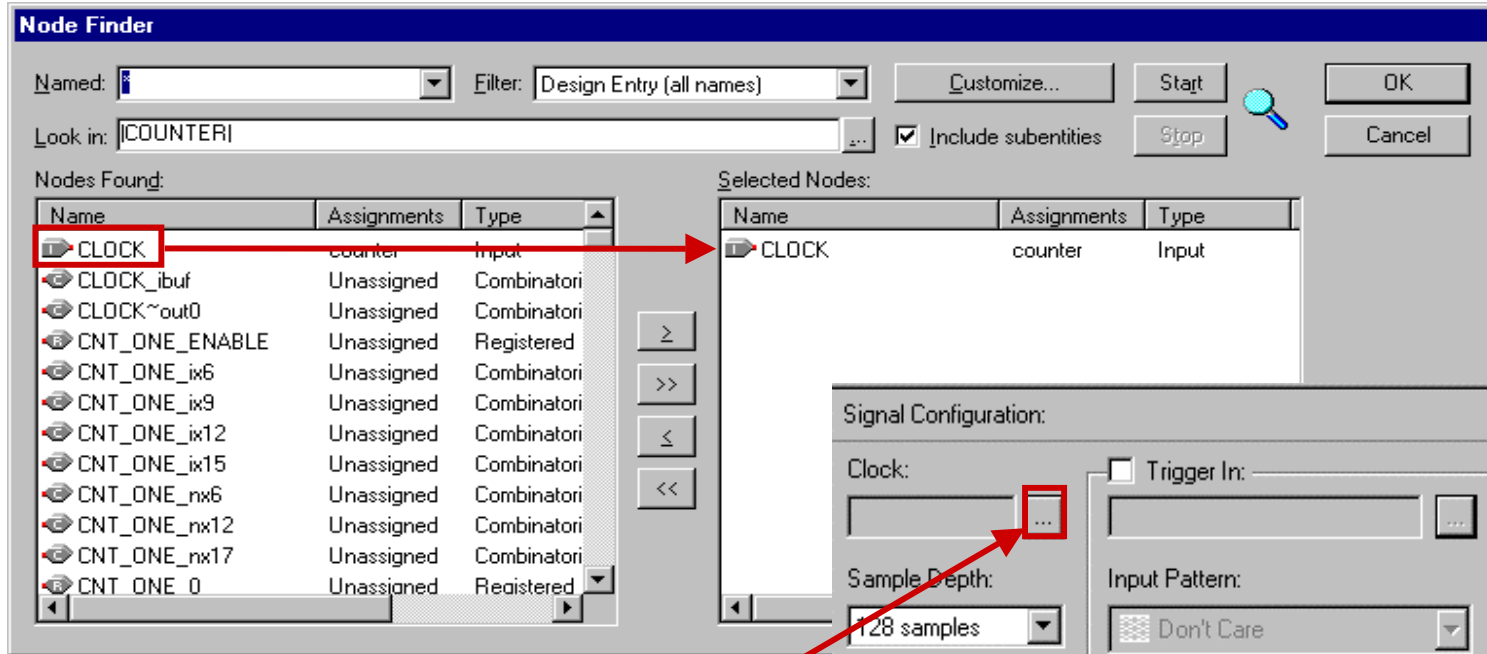
- Choose Save (File menu)

# Analyzer 2: Select Nodes

- Right Click in Instance Manager & Select Create Instance
- Open the Node Finder Window
- Click Start to List All Pins & Internal Nodes
- Add Following Signals
  - CNT\_ONE\_0, CNT\_ONE\_1, CNT\_ONE\_2, CNT\_ONE\_3
  - CNT\_ONE\_ENABLE
  - CNT\_TEN\_0, CNT\_TEN\_1, CNT\_TEN\_2, CNT\_TEN\_3
- Click OK

# Analyzer 2: Acquisition Clock

- Add Clock Input Pin from Node Finder



Open Node Finder for Clock

# Configure Analyzer 2

- Configure Logic Analyzer 2 (auto\_signaltap\_1) as Follows

128 Samples

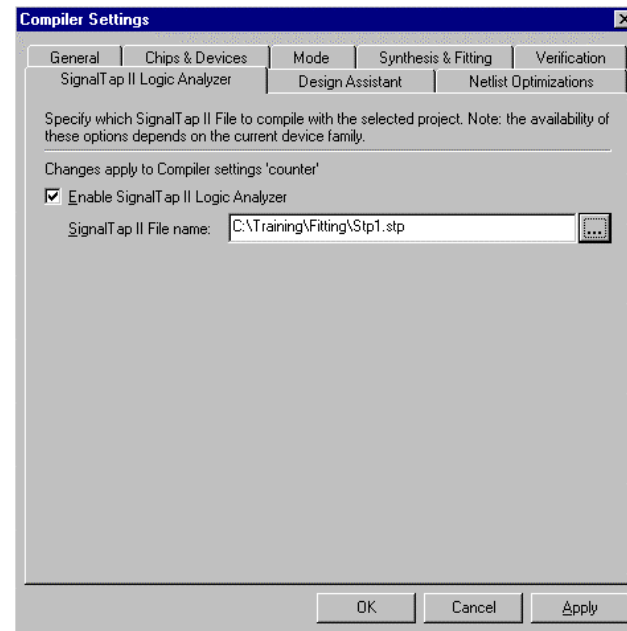
2 Levels

Center

- Choose Save (File menu)

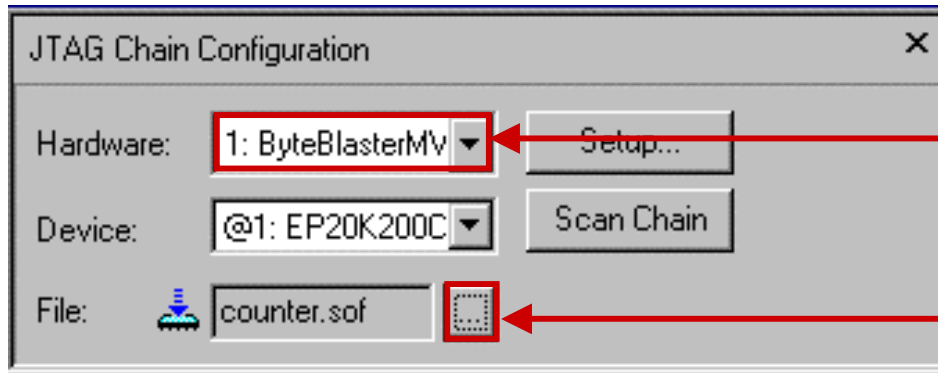
# Compile Design with SignalTap II Logic Analyzer

- Choose Compiler Settings (Processing Menu)
- Click the SignalTap II Logic Analyzer Tab
- Turn on Enable SignalTap II Logic Analyzer
- Browse to analysis.stp File
- Click OK
- Choose Start Compilation
- (Processing menu)



# Running Analysis

- Select Hardware & Configuration File



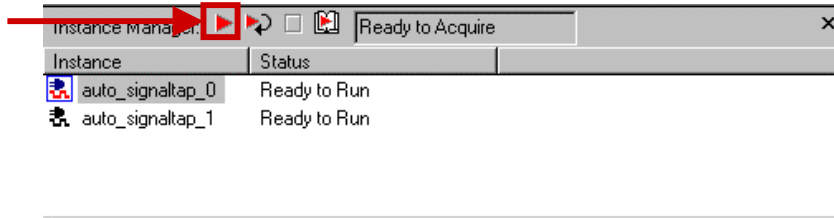
**Selects ByteBlasterMV Cable**

**Opens File Browser  
Select counter.sof**

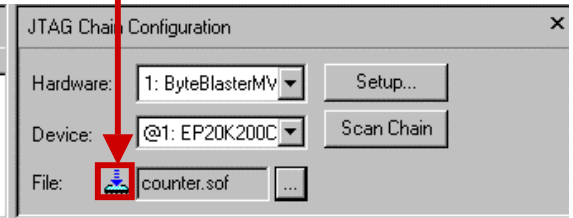
# Running Analyzer 1

- Set Trigger Pattern When CNT\_ONE Reaches 9
- Configure Device
- Run SignalTap II Analyzer

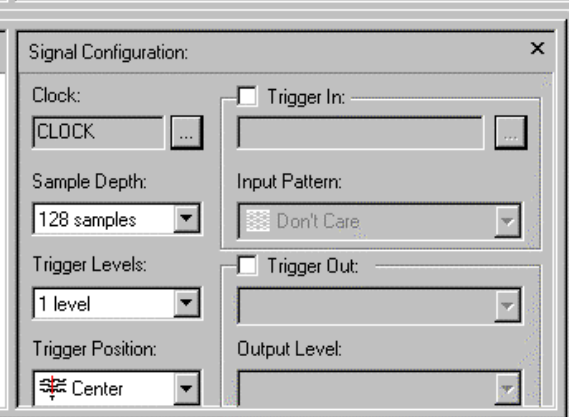
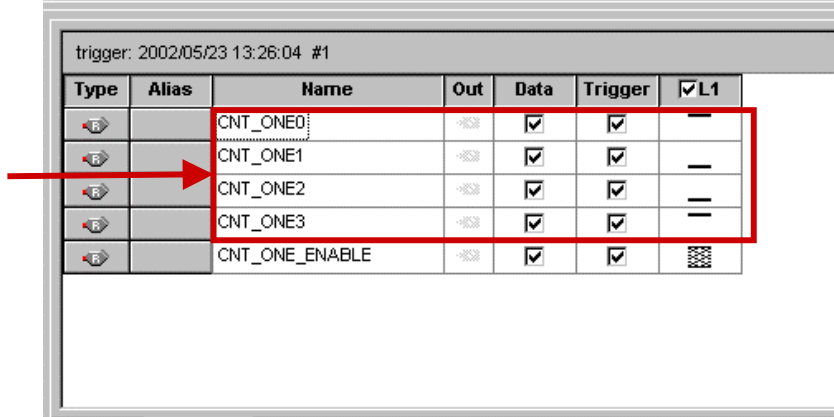
## 3. Run SignalTap II Analyzer



## 2. Configure Device



## 1. CNT\_ONE Value 9



# View Results

The screenshot shows the 'analysis.stp' window with the following components:

- Instance Manager:** A table with two instances, both 'Ready to Run':

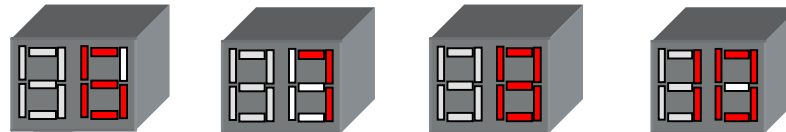
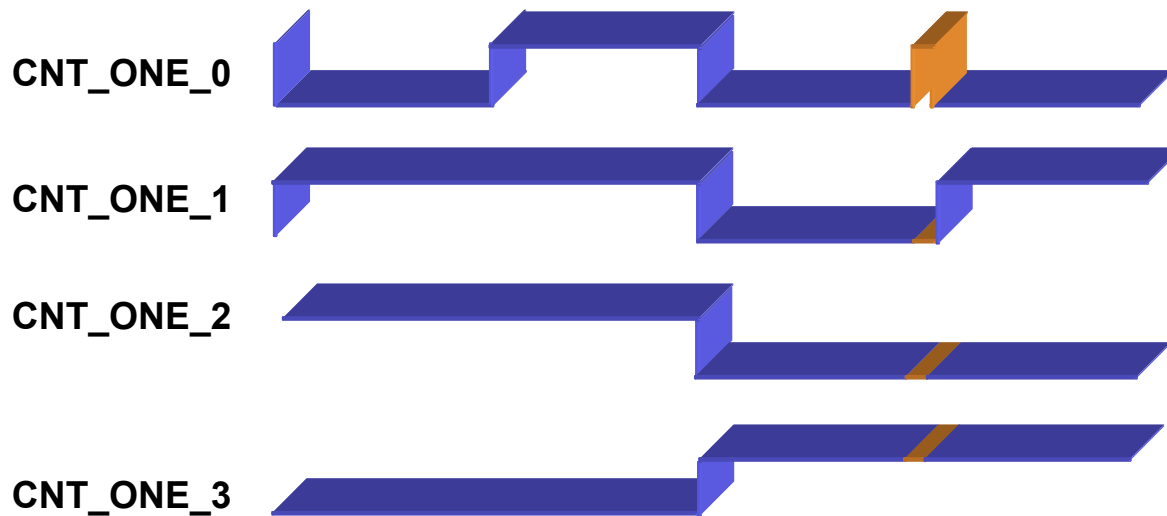
Instance	Status
auto_sigtaltap_0	Ready to Run
auto_sigtaltap_1	Ready to Run
- JTAG Chain Configuration:** Hardware: 1: ByteBlasterMv, Device: @1: EP20K200C, File: counter.sof.
- Signal Viewer:** A timing diagram showing signals CNT\_ONE0 through CNT\_ONE\_ENABLE. The time axis ranges from -16 to 112. A red box highlights the 'Data' tab in the viewer's bottom navigation bar.
- Hierarchy Display:** Shows a tree view with 'counter' selected.
- Data Log:** Shows 'auto\_sigtaltap\_0' as the active data source.

Results Displayed  
in Data Tab  
of Signal Viewer



# Isolating the Problem

- The CNT\_ONE Counts to 9 but its Synchronous Reset Signal Occurs on the Next Clock Immediately Resetting the Counter to Zero



# Analyzer 2 - Further Analysis

- Select Analyzer 2 - auto\_signaltap\_1

Double Click



The screenshot shows the 'analysis.stp' software interface. The 'Instance Manager' panel at the top left contains a table with two instances: 'auto\_signaltap\_0' and 'auto\_signaltap\_1', both with a status of 'Ready to Run'. A red box highlights the 'auto\_signaltap\_1' entry, with a red arrow pointing to it from the 'Double Click' text. The 'JTAG Chain Configuration' panel on the top right shows hardware '1: ByteBlasterMV', device '@1: EP20K200C', and file 'counter.sof'. The 'Signal Configuration' panel on the right shows clock 'CLOCK', sample depth '128 samples', trigger levels '2 levels', and trigger position 'Center'. The central panel displays a table of signals with columns for Type, Alias, Name, Out, Data, Trigger, L1, and L2. The 'Data' and 'Setup' tabs are visible at the bottom. The 'Hierarchy Display' panel at the bottom left shows a tree view with 'counter' selected. The 'Data Log' panel at the bottom right shows 'auto\_signaltap\_1' selected. A red box highlights the 'auto\_signaltap\_1' tab in the bottom navigation bar, with a red arrow pointing to it from the 'Select Tab' text.

Type	Alias	Name	Out	Data	Trigger	L1	L2
		CNT_ONE0	-	✓	✓	—	—
		CNT_ONE1	-	✓	✓	—	■
		CNT_ONE2	-	✓	✓	—	■
		CNT_ONE3	-	✓	✓	—	—
		CNT_ONE_ENABLE	-	✓	✓	■	■
		CNT_TEN0	-	✓	✓	—	—
		CNT_TEN1	-	✓	✓	—	—
		CNT_TEN2	-	✓	✓	—	—
		CNT_TEN3	-	✓	✓	—	—

Select Tab



# Running Analyzer 2

- Set Trigger Pattern As Shown
- Configure Device
- Run SignalTap II Analyzer

Run  
SignalTap II  
Analyzer

The screenshot displays the SignalTap II Analyzer interface. At the top, the 'Instance Manager' window shows two instances: 'auto\_signaltap\_0' and 'auto\_signaltap\_1', both with a status of 'Ready to Run'. A red arrow points to the 'Run' button in this window. To the right, the 'JTAG Chain Configuration' window is open, showing 'Hardware' set to '1: ByteBlasterMV', 'Device' set to '@1: EP20K200C', and 'File' set to 'counter.sof'. A red arrow points to the 'File' field. Below these, the 'Signal Configuration' window is open, showing 'Clock' set to 'CLOCK', 'Sample Depth' set to '128 samples', 'Trigger Levels' set to '2 levels', and 'Trigger Position' set to 'Center'. The 'Input Pattern' is set to 'Don't Care'. The main data table below shows the configuration for various signals. The 'L1' and 'L2' columns are highlighted with red boxes. The 'L1' column has a checkmark for 'CNT\_ONE0' through 'CNT\_TEN2' and a dash for 'CNT\_TEN3'. The 'L2' column has a dash for 'CNT\_ONE0' through 'CNT\_TEN2' and a checkmark for 'CNT\_TEN3'. The 'Trigger' column has a checkmark for all signals.

Type	Alias	Name	Out	Data	Trigger	✓L1	✓L2
		CNT_ONE0		✓	✓	—	—
		CNT_ONE1		✓	✓	—	—
		CNT_ONE2		✓	✓	—	—
		CNT_ONE3		✓	✓	—	—
		CNT_ONE_ENABLE		✓	✓	—	—
		CNT_TEN0		✓	✓	—	—
		CNT_TEN1		✓	✓	—	—
		CNT_TEN2		✓	✓	—	—
		CNT_TEN3		✓	✓	—	—

Configure  
Device

Trigger on 19 after Encountering 20 –

Notice that Analyzer Does Trigger on First Instance of 19

# View Results

The screenshot displays the 'analysis.stp' software interface. At the top, the 'Instance Manager' shows two instances: 'auto\_signaltap\_0' and 'auto\_signaltap\_1', both with a status of 'Ready to Run'. To the right, the 'JTAG Chain Configuration' panel shows hardware settings: '1: ByteBlasterMv', device '@1: EP20K200C', and file 'counter.sof'. The main area is a signal viewer showing a timing diagram with a time axis from -64 to 64. The table below lists the signals being monitored:

Type	Alias	Name
		CNT_ONE0
		CNT_ONE1
		CNT_ONE2
		CNT_ONE3
		CNT_ONE_ENABLE
		CNT_TEN0
		CNT_TEN1
		CNT_TEN2
		CNT_TEN3

Below the waveform, there are two tabs: 'Data' and 'Setup'. A red arrow points to the 'Data' tab, which is highlighted. The 'Data' tab shows a 'Hierarchy Display' with a checked box next to 'counter'. The 'Data Log' tab shows 'auto\_signaltap\_1'. At the bottom, the status bar shows 'auto\_signaltap\_0' and 'auto\_signaltap\_1'.

Results Displayed  
in Data Tab  
of Signal Viewer



**SOPC**  
**WORLD**  
2 0 0 2

## Conclusions

# SignalTap II Benefits

- Access Internal Signals within Design
- Easy Configuration through Quartus II Interface
- Using SignalTap II Analyzer Does Not Require Making Modifications to Design Files
- Available with Quartus II Software