Quartus II Software Design Series : Optimization

Optimization Techniques –

Timing Optimization

Timing Optimization

- General Recommendations
- Analyzing Timing Failures
- Solving Typical Timing Failures

General Recommendations

- Clocks
- I/O
- Asynchronous Control Signals

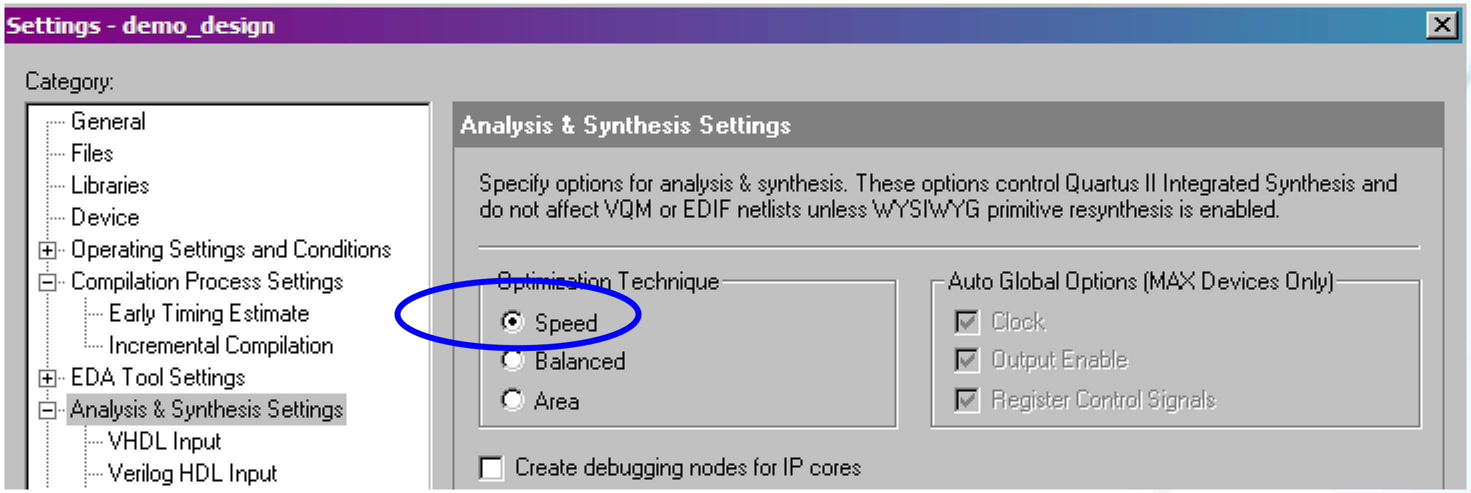
- Many of these suggestions are found in Timing Optimization Advisor & Quartus II Handbook

Clocks

- Optimize for Speed
  - Apply globally
  - Apply hierarchically
  - Apply to specific clock domain
- Enable netlist optimizations
- Enable physical synthesis

Global Speed Optimization

- Select speed
  - Default is balanced
  - Area-optimized designs may also show speed improvements
- May result in increased logic resource usage


Settings - demo_design

Category:

- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
  - Early Timing Estimate
  - Incremental Compilation
- EDA Tool Settings
- Analysis & Synthesis Settings
  - VHDL Input
  - Verilog HDL Input

Analysis & Synthesis Settings

Specify options for analysis & synthesis. These options control Quartus II Integrated Synthesis and do not affect VQM or EDIF netlists unless WYSIWYG primitive resynthesis is enabled.

Optimization Technique: [✓ Speed] Balanced Area

Auto Global Options (MAX Devices Only): Clock ✓, Output Enable ✓, Register Control Signals ✓

☐ Create debugging nodes for IP cores

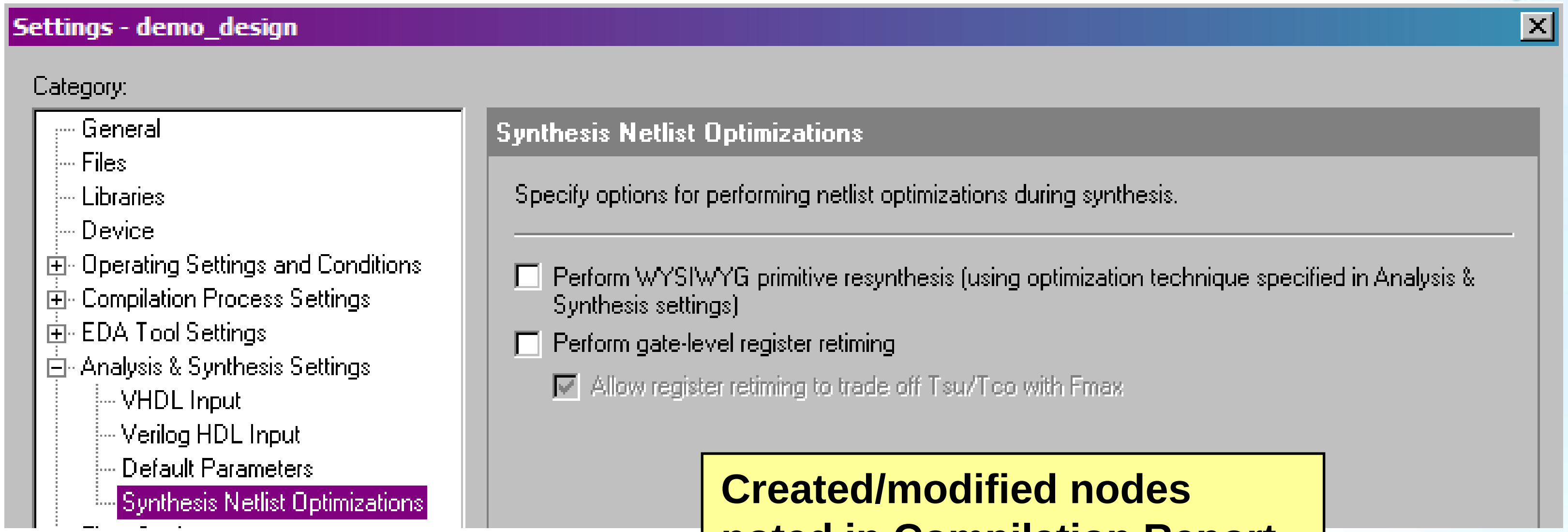
Individual Optimization

- Optimization Technique logic option
  - Use Assignment Editor or Tcl to apply to hierarchical block

- Speed Optimization Technique for Clock Domains logic option
  - Use Assignment Editor or Tcl to apply to clock domain or between clock domains

Synthesis Netlist Optimizations

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


Settings - demo_design
Category:
General
Files
Libraries
Device
Operating Settings and Conditions
Compilation Process Settings
EDA Tool Settings
Analysis & Synthesis Settings
VHDL Input
Verilog HDL Input
Default Parameters
Synthesis Netlist Optimizations

Synthesis Netlist Optimizations

Specify options for performing netlist optimizations during synthesis.

☐ Perform WYSIWYG primitive resynthesis (using optimization technique specified in Analysis & Synthesis settings)

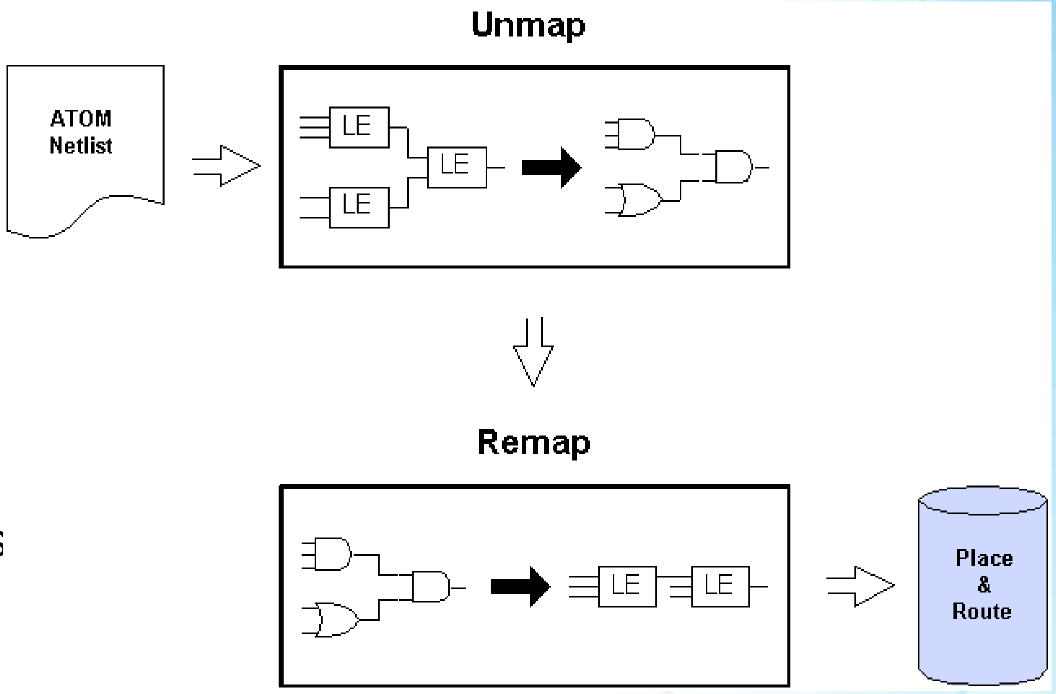
☐ Perform gate-level register retiming

✓ Allow register retiming to trade off Tsu/Tco with Fmax

Created/modified nodes noted in Compilation Report

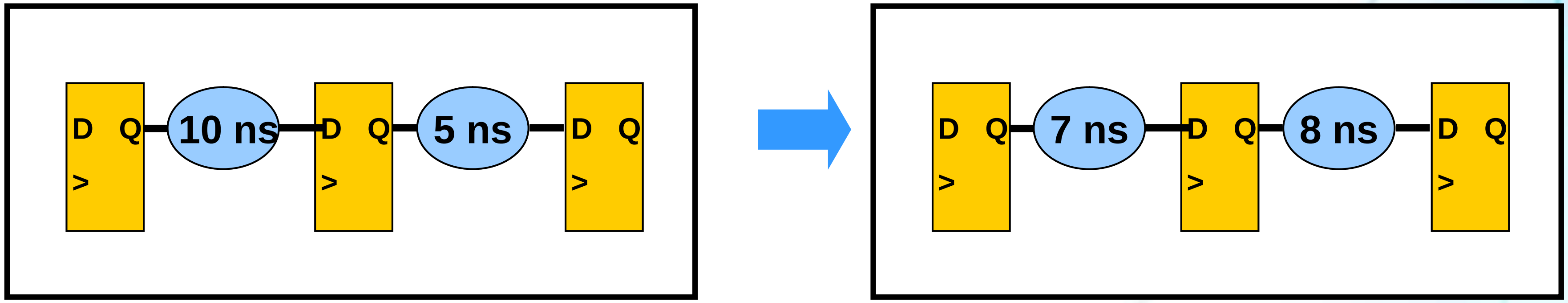
WYSIWYG Primitive Resynthesis

- Unmaps 3rd-party atom netlist back to gates & then remaps to Altera primitives
  - Not intended for use with integrated synthesis
- Considerations
  - Node names may change
  - 3rd-party synthesis attributes may be lost
    - Preserve/keep
  - Some registers may be synthesized away



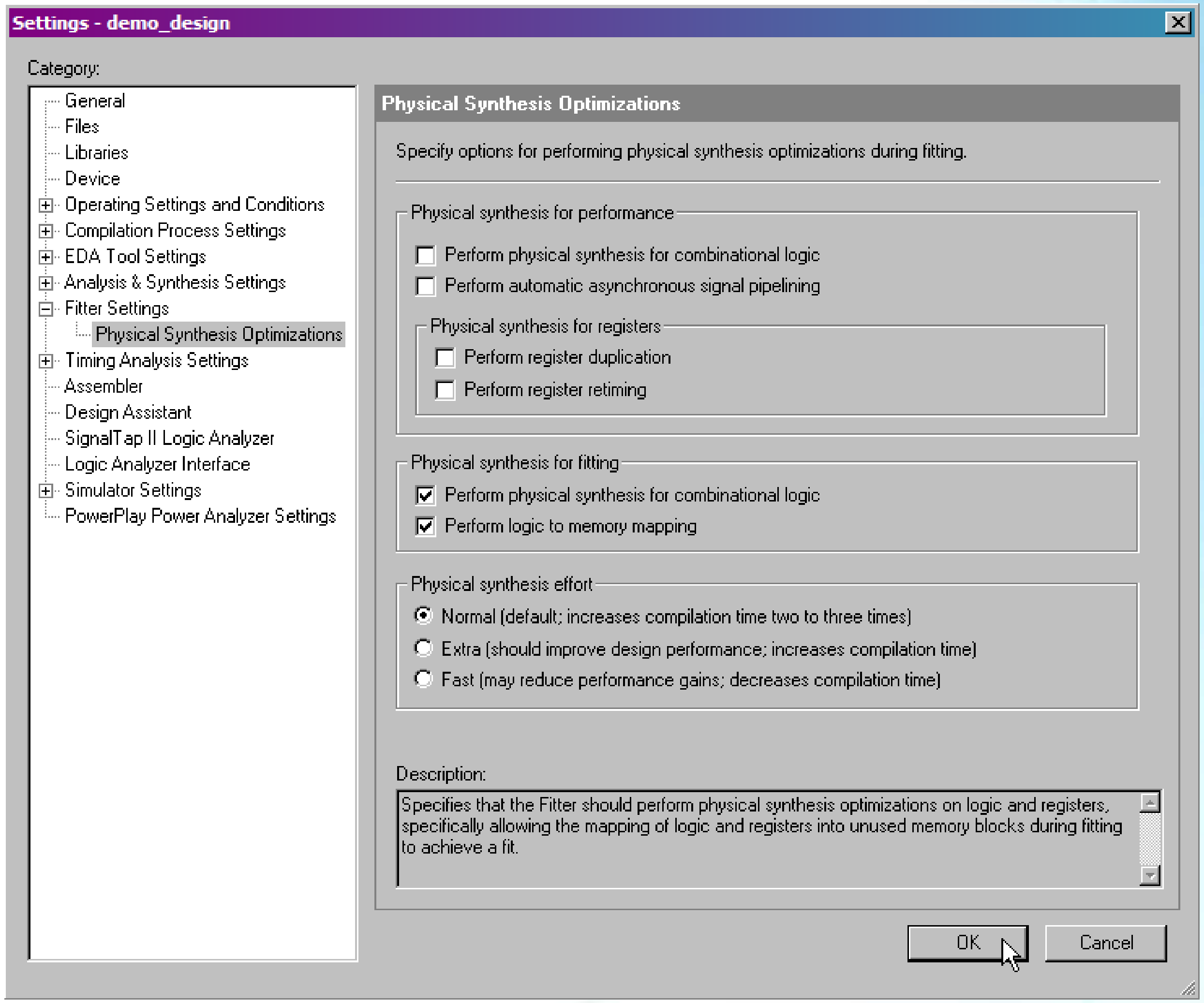
Gate-Level Register Retiming

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


Settings - demo_design

Category:
- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
- EDA Tool Settings
- Analysis & Synthesis Settings
- Fitter Settings
  - Physical Synthesis Optimizations
- Timing Analysis Settings
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- Simulator Settings
- PowerPlay Power Analyzer Settings

Physical Synthesis Optimizations

Specify options for performing physical synthesis optimizations during fitting.

Physical synthesis for performance
- [ ] Perform physical synthesis for combinational logic
- [ ] Perform automatic asynchronous signal pipelining

Physical synthesis for registers
- [ ] Perform register duplication
- [ ] Perform register retiming

Physical synthesis for fitting
- [✓] Perform physical synthesis for combinational logic
- [✓] Perform logic to memory mapping

Physical synthesis effort
- ● Normal (default; increases compilation time two to three times)
- ○ Extra (should improve design performance; increases compilation time)
- ○ Fast (may reduce performance gains; decreases compilation time)

Description:
Specifies that the Fitter should perform physical synthesis optimizations on logic and registers, specifically allowing the mapping of logic and registers into unused memory blocks during fitting to achieve a fit.

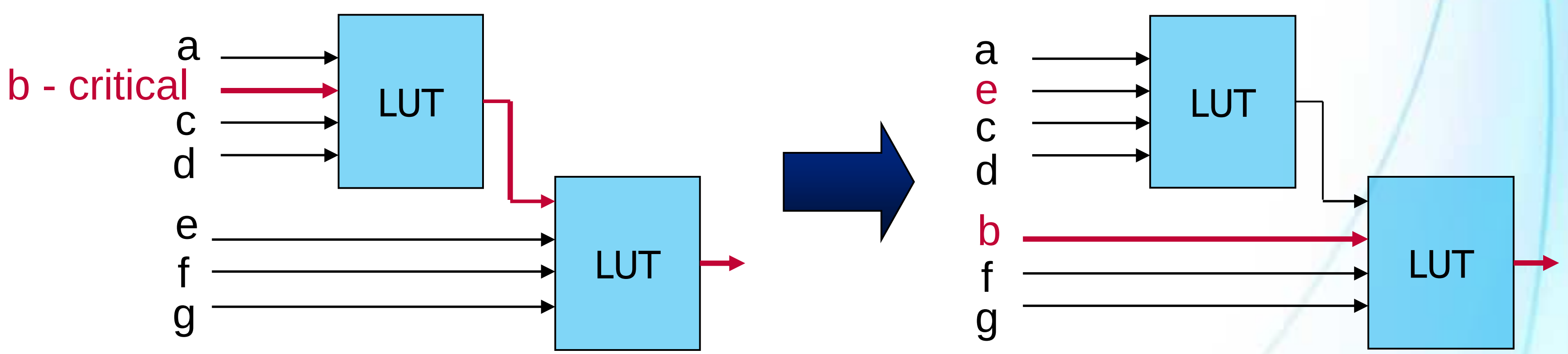
OK Cancel

Physical Synthesis

- Types
  - Targeting performance:
    - Combinational logic
    - Asynchronous signal pipelining
    - Register duplication
    - Register retiming
  - Targeting fitting
    - Physical synthesis for combinatorial logic
    - Logic to memory mapping
- Effort
  - Trades performance vs. compile time
  - Normal, extra or fast
- New or modified nodes appear in Compilation Report

Combinational Logic

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

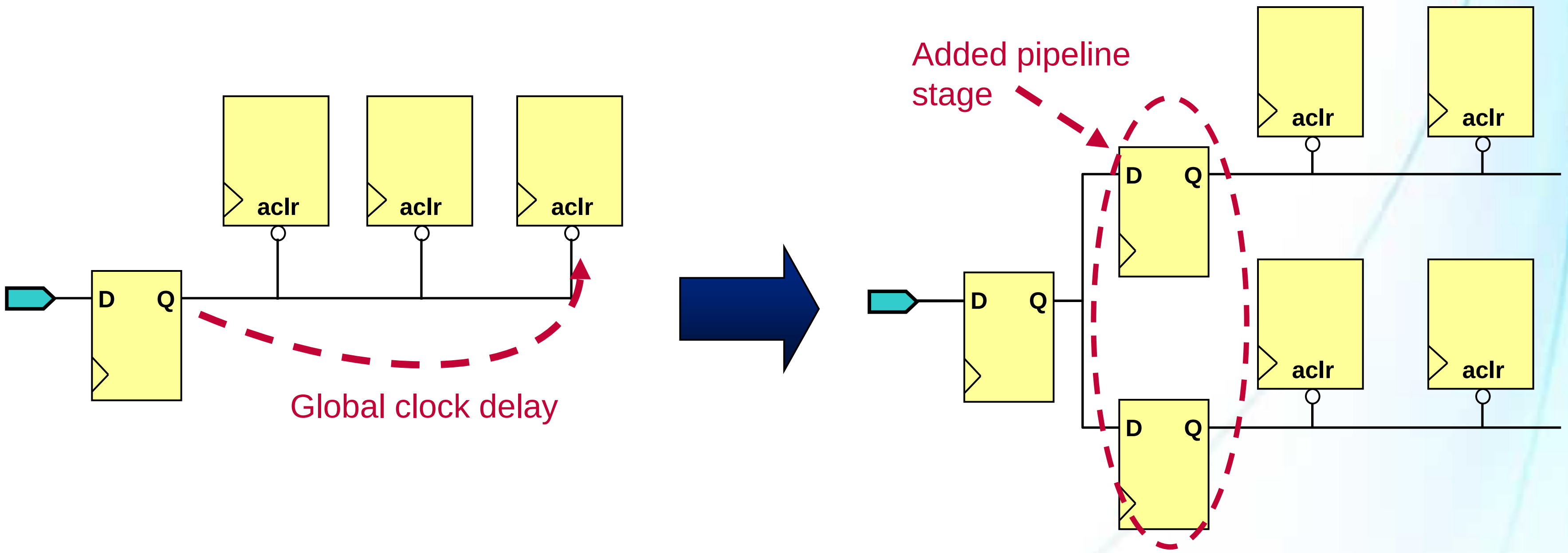


Asynchronous Control Signals

- Improve Recovery & Removal Timing
- Make control signal non-global
  - Project-wide
    - Assignments ⇒ Settings ⇒ Fitter Settings ⇒ More Settings
  - Individually
    - Set Global Signal logic option to Off
- Enable “Automatic asynchronous signal pipelining” option (physical synthesis)

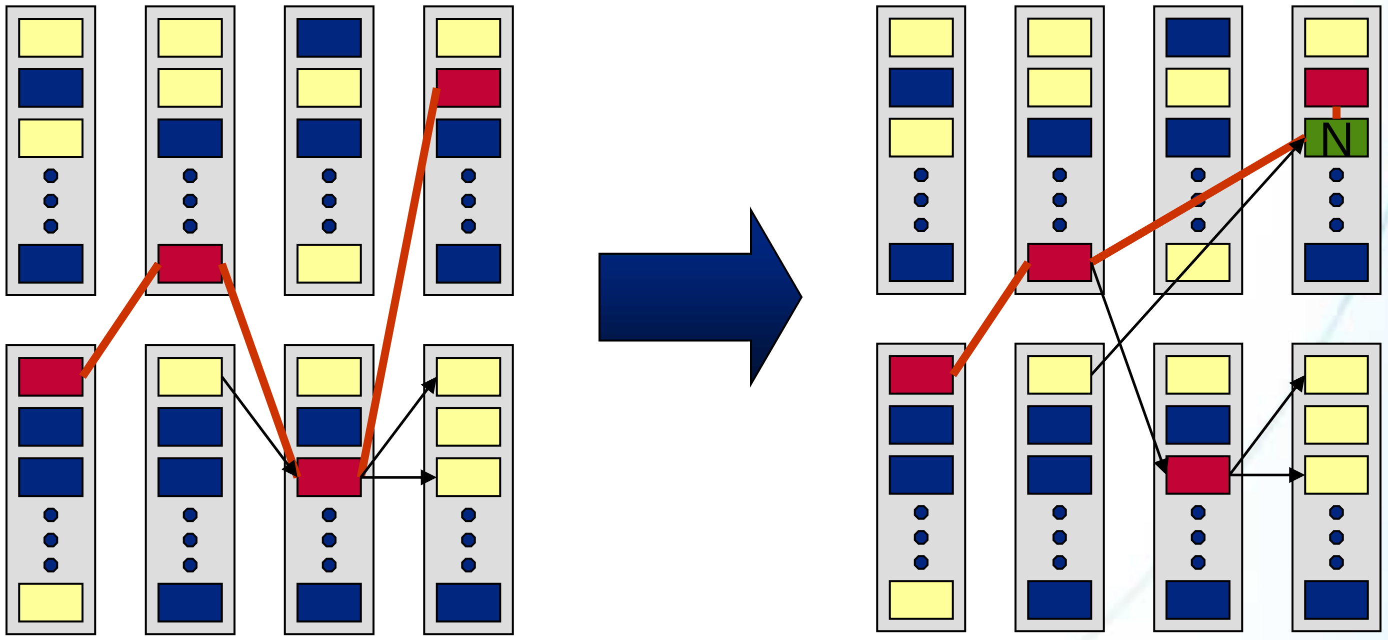
Asynchronous Signal Pipelining

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



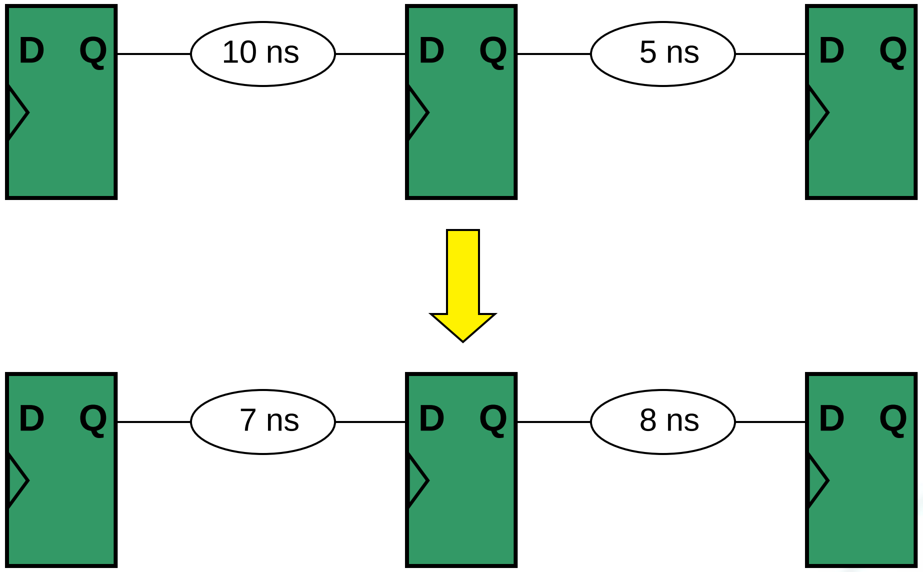
Duplication

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



Register Retiming

- Uses fewer registers than pipelining
  - Trade off the delay between timing-critical and non-critical paths
  - Reduce switching
  - Does not change logic functionality

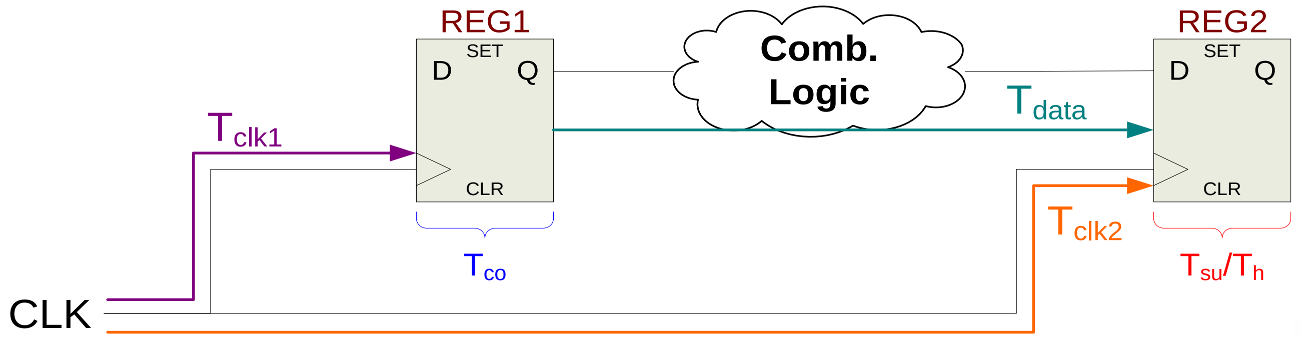


Timing Optimization

- General Recommendations
- Analyzing Timing Failures
- Solving Typical Timing Failures

Analyzing Timing Failures

- Typical synchronous path
  - Registers can be internal or external to FPGA



	REG1	REG2
Input Failure	**External**	**Internal**
Output Failure	**Internal**	**External**
Failure within Clock Domain	**Internal**	**Internal**

Slack Equations

Setup Slack Equation:

$$\underbrace{(\text{latch edge} + T_{clk2} - T_{su})}_{\text{Data Required}} - \underbrace{(\text{launch edge} + T_{clk1} + T_{co} + T_{data})}_{\text{Data Arrival}}$$

T_{su}, T_h, T_{co} are usually fixed values; Function of silicon

Hold Slack Equation:

$$\underbrace{(\text{launch edge} + T_{clk1} + T_{co} + T_{data})}_{\text{Data Arrival}} - \underbrace{(\text{latch edge} + T_{clk2} + T_h)}_{\text{Data Required}}$$

Slack Equations (cont.)

Setup Slack Equation:

$$\underbrace{(\text{latch edge} + T_{clk2} - T_{su})}_{\text{Data Required}} - \underbrace{(\text{launch edge} + T_{clk1} + T_{co} + T_{data})}_{\text{Data Arrival}}$$

Timing issues show up here

Hold Slack Equation:

$$\underbrace{(\text{launch edge} + T_{clk1} + T_{co} + T_{data})}_{\text{Data Arrival}} - \underbrace{(\text{latch edge} + T_{clk2} + T_{h})}_{\text{Data Required}}$$

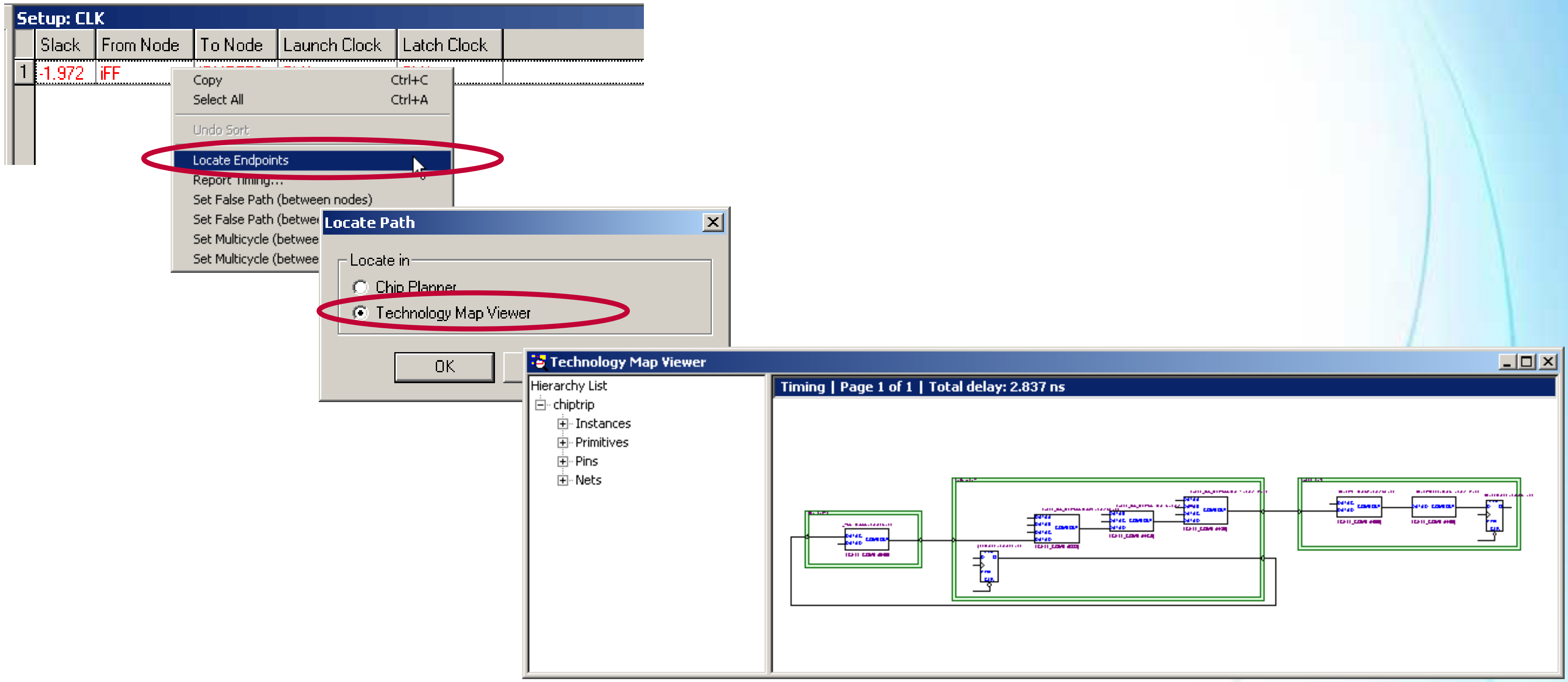
Typical Timing Errors

- Clock delays (T_{clk1} or T_{clk2})
  - Ripple/gated clocks
  - Non-global routing
- Data path delay (T_{data})
  - Fan-out
  - Too many logic levels
  - Poor placement
  - Physical limitations

Exploring Failures in Quartus II Software

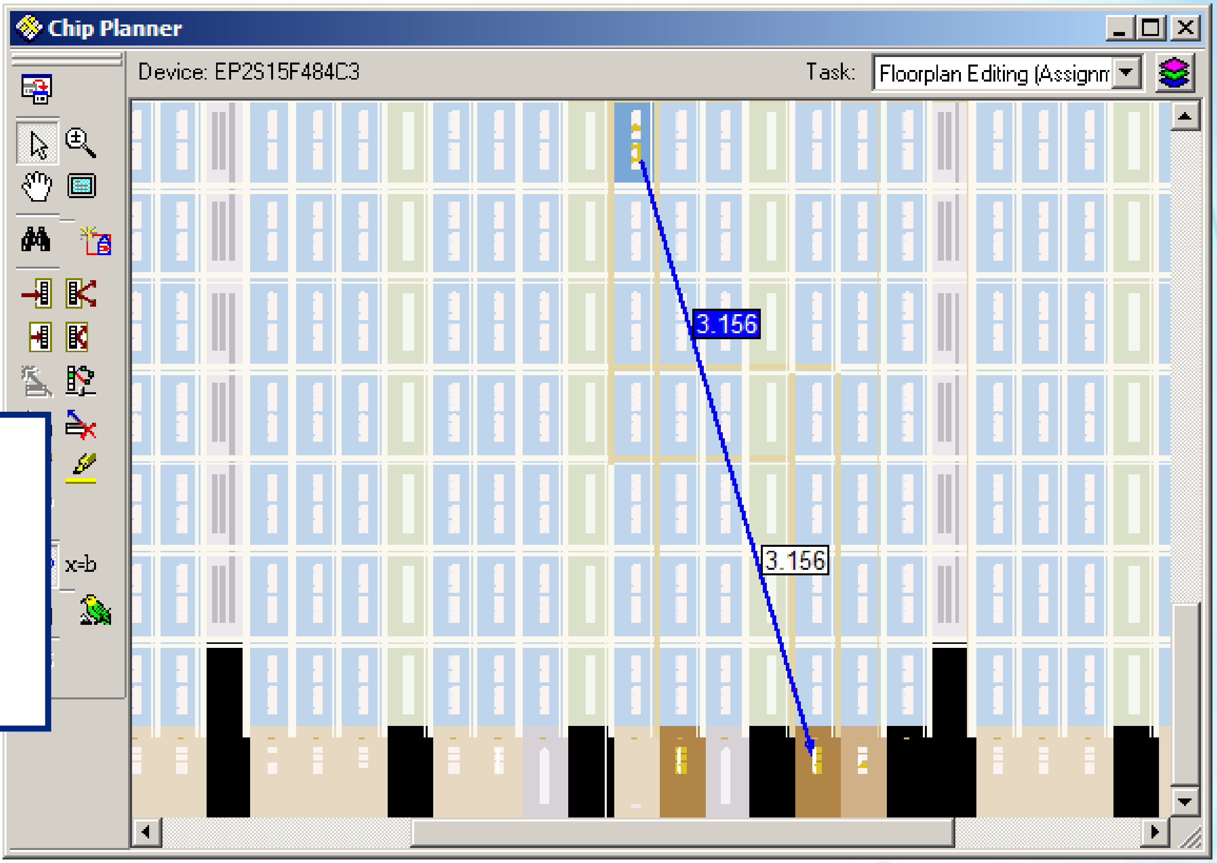
- Technology Map Viewer
  - Graphically shows number of logic levels
- Chip Planner
  - Graphically shows placement
- TimeQuest path analysis
  - Highlights clock/path delays
  - Highlights fan-out
  - Highlights number of logic levels
  - And just about everything else

Technology Map Viewer



- Accessing Technology Map Viewer
  - Right-click in TimeQuest report and choose Locate Path or Locate Endpoints
- View number of logic levels in failing paths

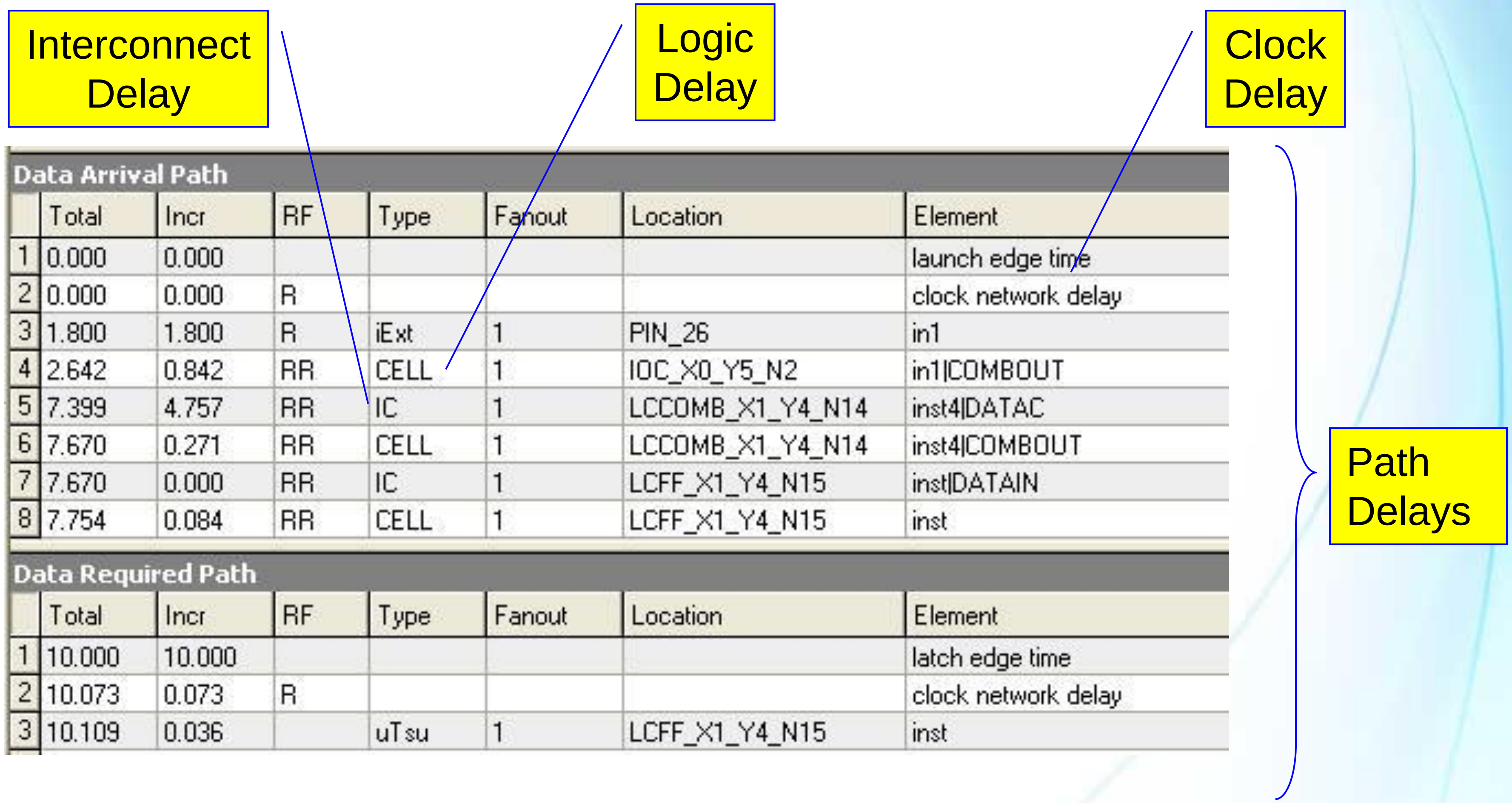
Chip Planner



Choose Chip Planner and the Chip Planner displays the placement, routing & timing information for that path

- Accessing Chip Planner
  - Right-click in TimeQuest report and choose Locate Path or Locate Endpoints
- View placement of nodes in timing path as well as chosen routing

TimeQuest Path Analysis



Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	0.000	0.000	R				clock network delay
3	1.800	1.800	R	iExt	1	PIN_26	in1
4	2.642	0.842	RR	CELL	1	IOC_X0_Y5_N2	in1|COMBOUT
5	7.399	4.757	RR	IC	1	LCCOMB_X1_Y4_N14	inst4|DATAC
6	7.670	0.271	RR	CELL	1	LCCOMB_X1_Y4_N14	inst4|COMBOUT
7	7.670	0.000	RR	IC	1	LCFF_X1_Y4_N15	inst|DATAIN
8	7.754	0.084	RR	CELL	1	LCFF_X1_Y4_N15	inst

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	10.000	10.000					latch edge time
2	10.073	0.073	R				clock network delay
3	10.109	0.036		uTsu	1	LCFF_X1_Y4_N15	inst

- Provides ALL detailed information pertaining to timing path

Further Path Analysis

- Always start with worst slack path(s)
  - Fixing worst path(s) may give Fitter freedom to fix other failing paths
- In TimeQuest reports, list top 50-100 failing paths and look for common source, intermediate or destination nodes
  - Sometimes start or end nodes are bits of same bus
  - Sometimes paths with different source or destination nodes have common intermediate nodes

Timing Optimization

- General Recommendations
- Analyzing Timing Failures
- <u>Solving Typical Timing Failures</u>

Solving Typical Timing Failures

We'll look at some cases of timing failures, how to identify them and possible solutions. It is possible for you to have several at once.

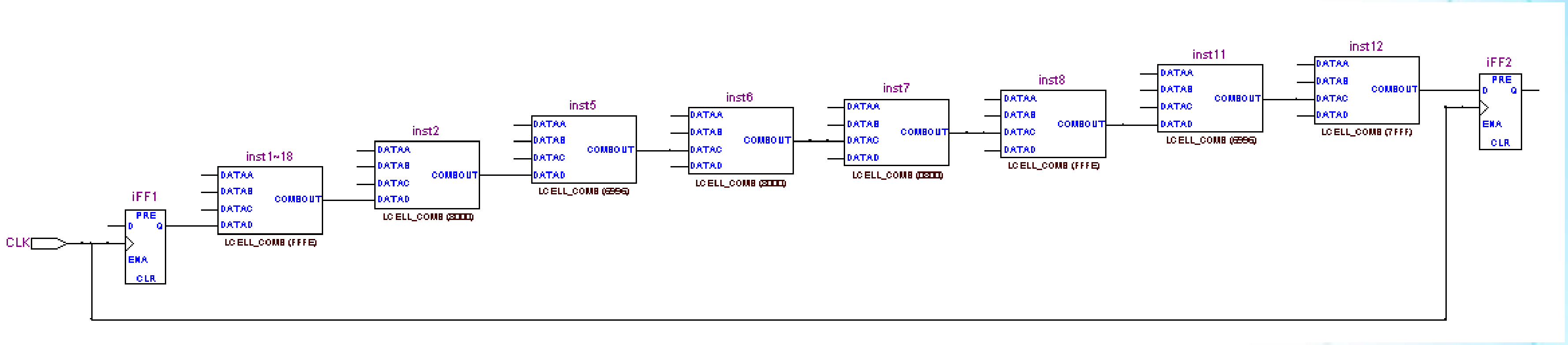
1) Too many logic levels
2) Fan-out signals
3) Conflicting physical constraints
4) Conflicting timing assignments
5) Tight timing requirements

Case 1) Too Many Logic Levels

- Increases T_{data}, thus increasing data arrival time
- How to verify
  - Technology Map Viewer on failing path
  - TimeQuest detailed path analysis

Case 1) Technology Map Viewer

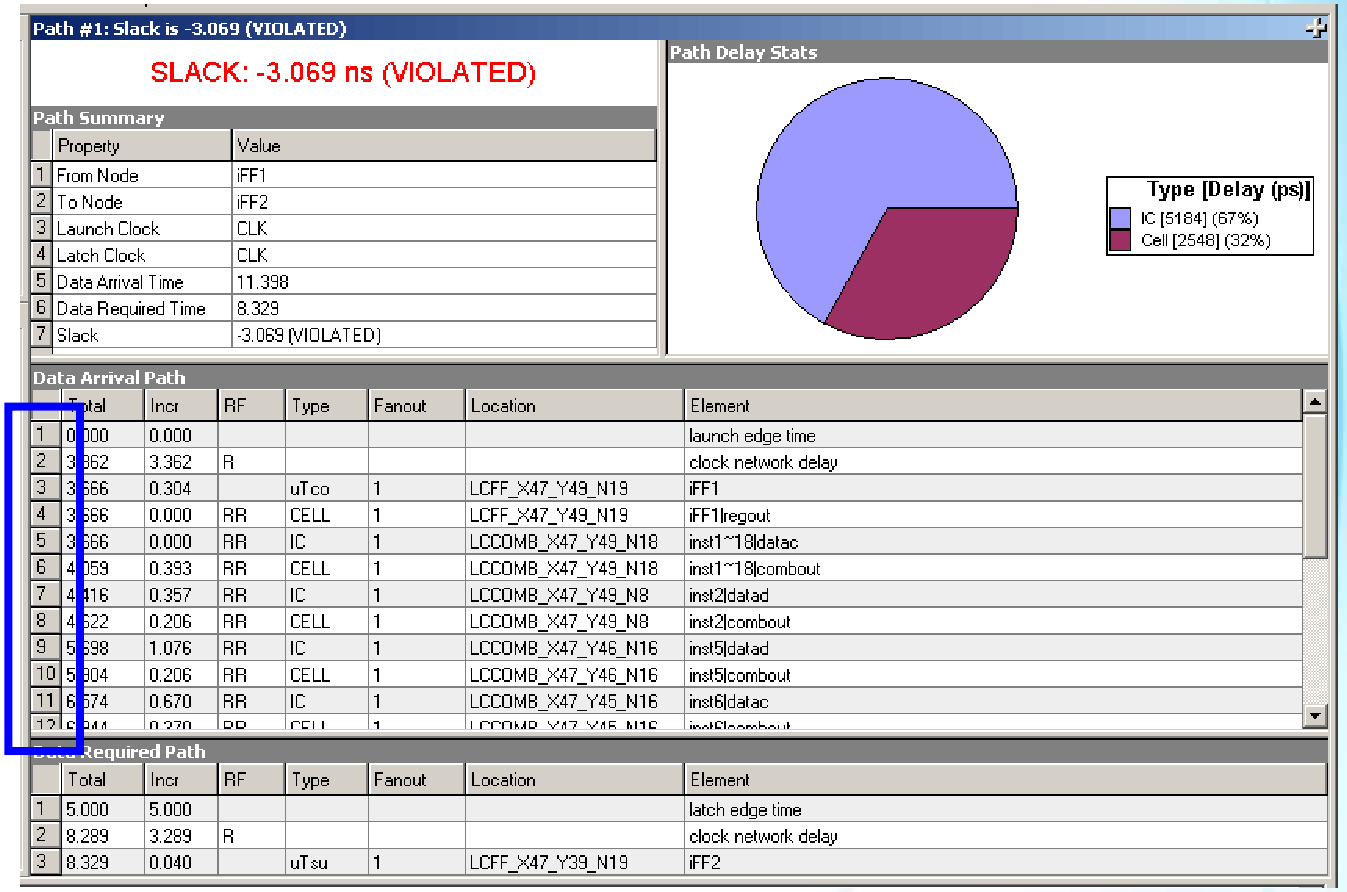
Right-click on failing path and select Locate Endpoints or Path



This path has 8 levels of logic

Case 1) TimeQuest

Note number of levels of logic in data arrival path



Path #1: Slack is -3.069 (VIOLATED)

SLACK: -3.069 ns (VIOLATED)

Path Summary

	Property	Value
1	From Node	iFF1
2	To Node	iFF2
3	Launch Clock	CLK
4	Latch Clock	CLK
5	Data Arrival Time	11.398
6	Data Required Time	8.329
7	Slack	-3.069 (VIOLATED)

Path Delay Stats

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	3.362	3.362	R				clock network delay
3	3.666	0.304		uTco	1	LCFF_X47_Y49_N19	iFF1
4	3.666	0.000	RR	CELL	1	LCFF_X47_Y49_N19	iFF1|regout
5	3.666	0.000	RR	IC	1	LCCOMB_X47_Y49_N18	inst1~18|datac
6	4.059	0.393	RR	CELL	1	LCCOMB_X47_Y49_N18	inst1~18|combout
7	4.416	0.357	RR	IC	1	LCCOMB_X47_Y49_N8	inst2|datad
8	4.622	0.206	RR	CELL	1	LCCOMB_X47_Y49_N8	inst2|combout
9	5.698	1.076	RR	IC	1	LCCOMB_X47_Y46_N16	inst5|datad
10	5.904	0.206	RR	CELL	1	LCCOMB_X47_Y46_N16	inst5|combout
11	6.574	0.670	RR	IC	1	LCCOMB_X47_Y45_N16	inst6|datac
12	6.844	0.270	RR	CELL	1	LCCOMB_X47_Y45_N16	inst6|combout

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	5.000	5.000					latch edge time
2	8.289	3.289	R				clock network delay
3	8.329	0.040		uTsu	1	LCFF_X47_Y39_N19	iFF2

Case 1) Possible Solutions

- Add multi-cycle assignments if design allows

$\}$ Changes Launch & Latch Edges

- Add pipeline registers
  - Reduces logic levels
  - Adds latency

$\}$ Changes T_{data}

- Enable register retiming (physical synthesis)
  - Redistributes logic around registers reducing number of levels
  - Increases compile time

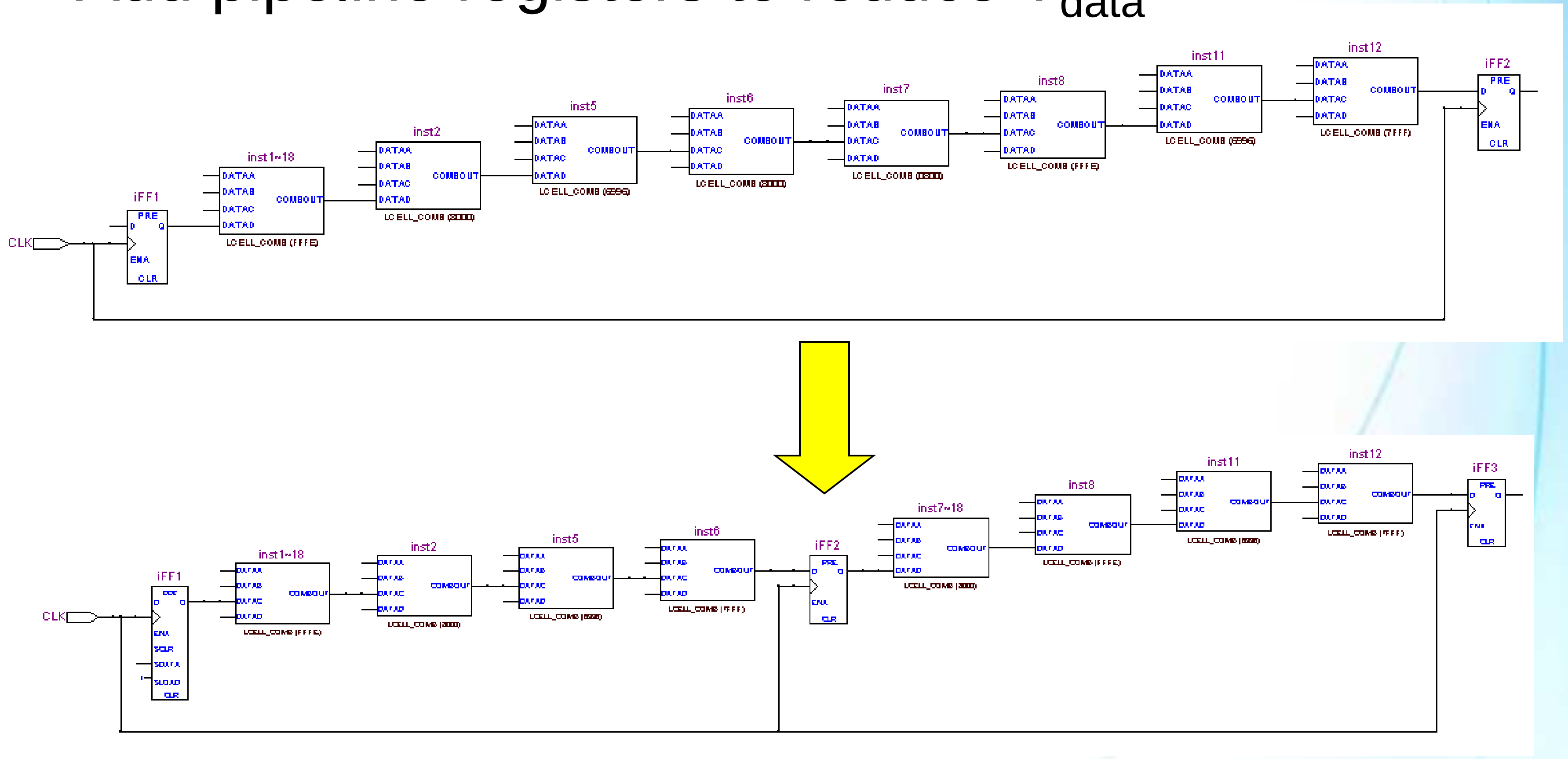
$\}$ Changes T_{data}

- Recode logic to be more efficient
  - Reduces logic levels
  - May need to focus on implementation

$\}$ Changes T_{data}

Case 1) Pipeline Registers

- Add pipeline registers to reduce T_{data}



Case 1) Focus on Implementation

- HDL coding decisions will **greatly** impact resulting synthesis
  - May need to code with resulting synthesis in mind

- See Quartus II handbook chapter, “Recommended HDL Coding Styles”

- Great material on HDL coding

Tip #1 - Reduce Embedded IFs

- Don't embed IF statements
  - Use CASE statements instead

VHDL

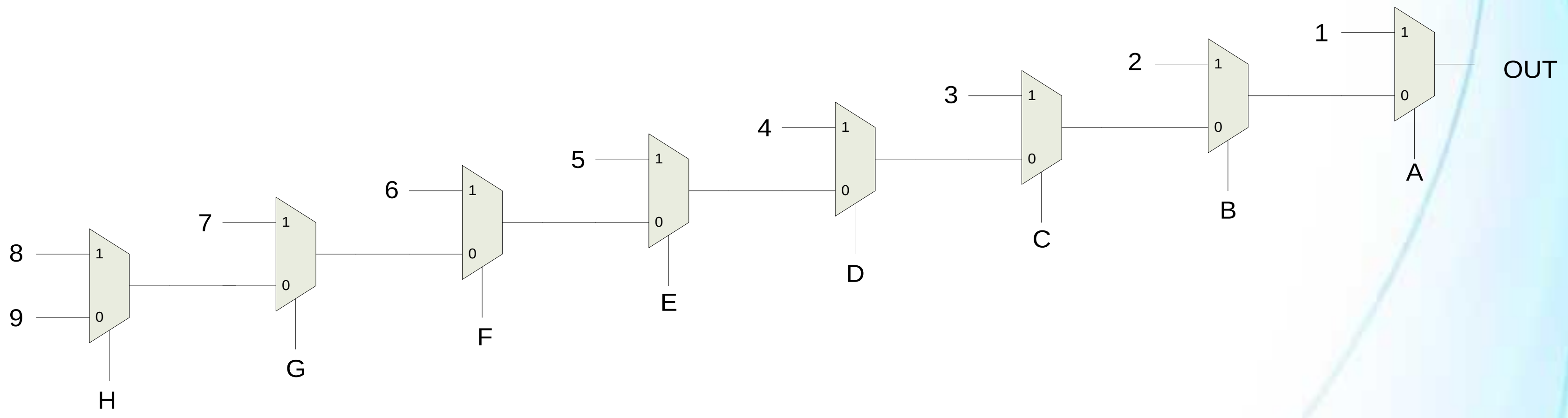
```
-- Too many embedded IF statements
process(A, B, C, D, E, F, G, H)
begin
   if A = '1' then
      sig_out <= 1;
   elsif B = '1' then
      sig_out <= 2;
   elsif C = '1' then
      sig_out <= 3;
   elsif D = '1' then
      sig_out <= 4;
   elsif E = '1' then
      sig_out <= 5;
   elsif F = '1' then
      sig_out <= 6;
   elsif G = '1' then
      sig_out <= 7;
   elsif H = '1' then
      sig_out <= 8;
   else
      sig_out <= 9;
   end if;
end process;
```

Verilog

```
// Too many embedded IF statements
always @(*)
begin
   if (A)
      sig_out <= 1;
   else if (B)
      sig_out <= 2;
   else if (C)
      sig_out <= 3;
   else if (D)
      sig_out <= 4;
   else if (E)
      sig_out <= 5;
   else if (F)
      sig_out <= 6;
   else if (G)
      sig_out <= 7;
   else if (H)
      sig_out <= 8;
   else
      sig_out <= 9;
end
```

Tip #1 - Reduce Embedded IFs (cont.)

- Resulting hardware interpretation



Tip #2 - Use System Verilog Unique Case

- Verilog CASE implies one-to-many relationship
- Verilog CASE statement is implemented as a priority encoder
  - i.e. embedded IF statements
- System Verilog is a superset of Verilog
- Use “unique” qualifier to prevent priority encoder

Unique and Priority

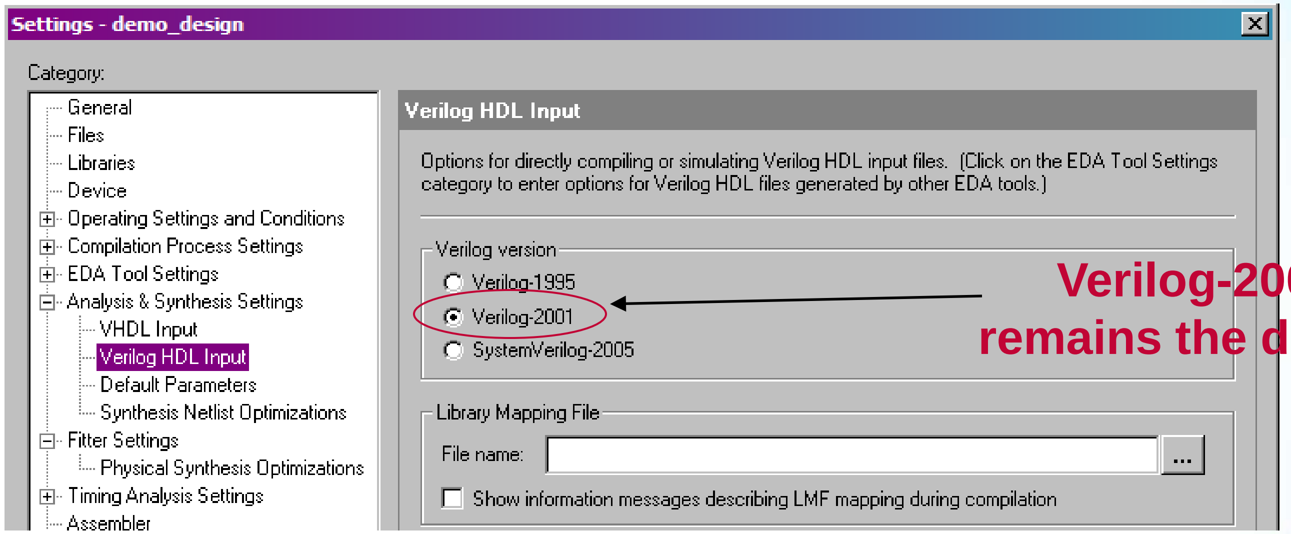
- `unique` and `priority` keywords apply to case statements or if/else chains
- `unique` implies non-overlapping case items or conditional expressions
- `priority` implies just the opposite

```
unique case(state)
S0:
S1:
S2:
endcase
```

No more parallel_case!

Enabling SystemVerilog-2005

- GUI



Settings - demo_design

Category:

- General
- Files
- Libraries
- Device
- Operating Settings and Conditions
- Compilation Process Settings
- EDA Tool Settings
- Analysis & Synthesis Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
  - Synthesis Netlist Optimizations
- Fitter Settings
  - Physical Synthesis Optimizations
- Timing Analysis Settings
- Assembler

Verilog HDL Input

Options for directly compiling or simulating Verilog HDL input files. (Click on the EDA Tool Settings category to enter options for Verilog HDL files generated by other EDA tools.)

Verilog version

- ○ Verilog-1995
- ● Verilog-2001
- ○ SystemVerilog-2005

Library Mapping File

File name: ...

☐ Show information messages describing LMF mapping during compilation

Verilog-2001 remains the default

- Source-level control (for IP etc)

```
// synthesis VERILOG_INPUT_VERSION SYSTEMVERILOG_2005

module(input byte a, b, output logic);
```

- Per-file basis

```
set_global_assignment -name VERILOG_FILE -rev SYSTEMVERILOG_2005
```

Tip #3: CASE synthesis directives

- Don’t use synthesis directives
  - parallel_case
  - full_case

- Great paper discusses the perils of CASE synthesis directives
  - "full_case parallel_case",the Evil Twins of Verilog Synthesis
    - (http://www.sunburst-design.com/papers/CummingsSNUG1999Boston_FullParallelCase.pdf)

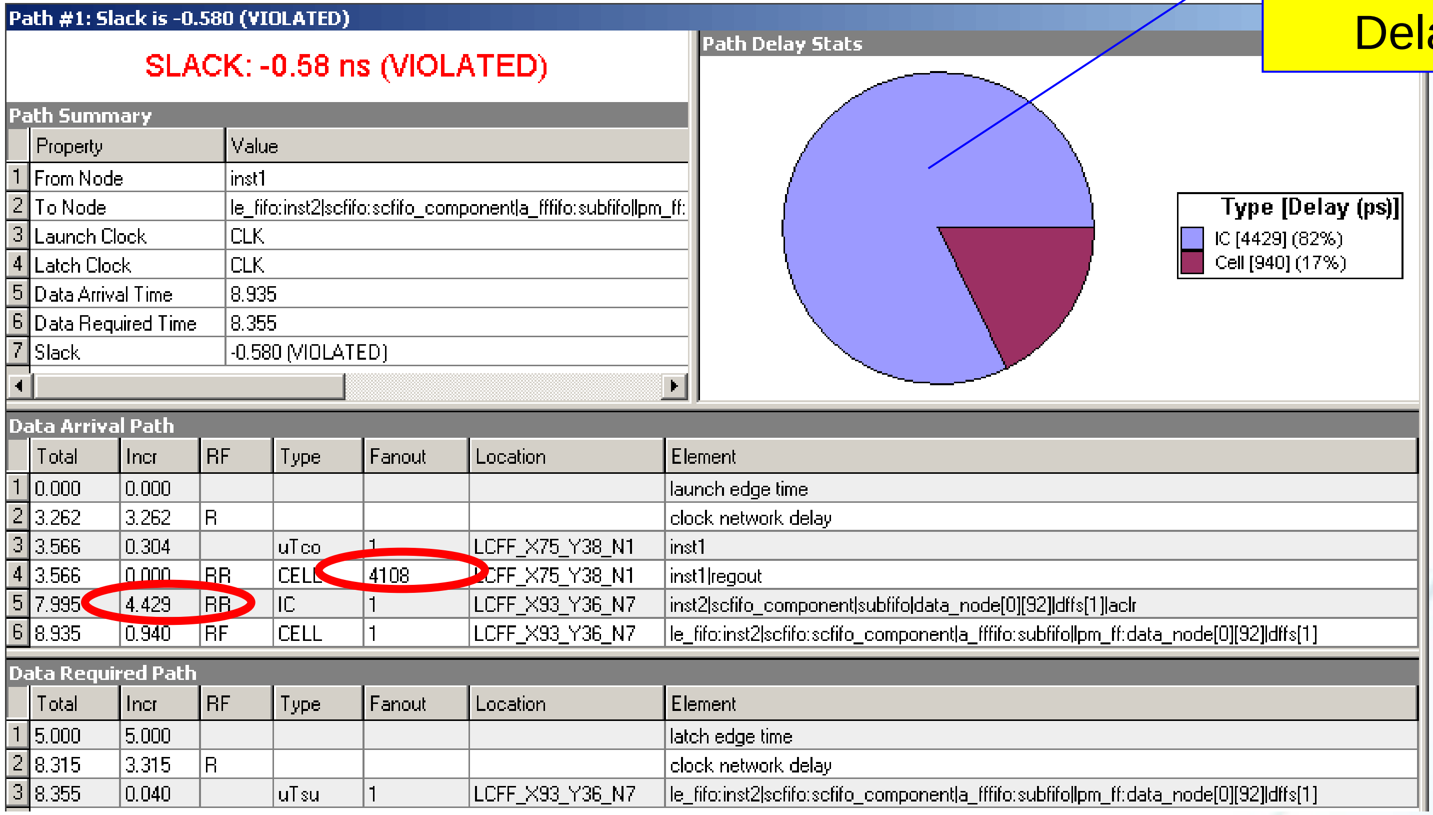
Case 2) Fan-Out Signals

- Timing failures from fan-out are more often a matter of **where** than of **how many**
  - High fan-out in itself can force nodes to spread out or can result in slow routing
    - Increases routing delay and thus T_{data}
    - Proximity is key in FPGAs & newer CPLDs
- Typical problem cases:
  - Memory control signals
  - Clock enables

Case 2) Fan-Out Signals (cont.)

- How to verify
  - Locate high fan-out signals as possible causes
    - TimeQuest path analysis
    - Non-Global High Fan-Out Signals table in Compilation Report (Fitter folder ⇒Resource section)
  - Use Chip Planner to verify locations of nodes

Case 2) Timequest



Interconnect Delay

Path #1: Slack is -0.580 (VIOLATED)

SLACK: -0.58 ns (VIOLATED)

Path Summary

	Property	Value
1	From Node	inst1
2	To Node	le_fifo:inst2|scfifo:scfifo_component|a_fffifo:subfifo|lpm_ff:
3	Launch Clock	CLK
4	Latch Clock	CLK
5	Data Arrival Time	8.935
6	Data Required Time	8.355
7	Slack	-0.580 (VIOLATED)

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	3.262	3.262	R				clock network delay
3	3.566	0.304		uTco	1	LCFF_X75_Y38_N1	inst1
4	3.566	0.000	RR	CELL	4108	LCFF_X75_Y38_N1	inst1|regout
5	7.995	4.429	RR	IC	1	LCFF_X93_Y36_N7	inst2|scfifo_component|subfifo|data_node[0][92]|dffs[1]|aclr
6	8.935	0.940	RF	CELL	1	LCFF_X93_Y36_N7	le_fifo:inst2|scfifo:scfifo_component|a_fffifo:subfifo|lpm_ff:data_node[0][92]|dffs[1]

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	5.000	5.000					latch edge time
2	8.315	3.315	R				clock network delay
3	8.355	0.040		uTsu	1	LCFF_X93_Y36_N7	le_fifo:inst2|scfifo:scfifo_component|a_fffifo:subfifo|lpm_ff:data_node[0][92]|dffs[1]

Fanout of 4108 with interconnect delay of 4.429 ns

Case 2) Possible Solutions

- Add multi-cycle assignments if design allows

$\}$ Changes Launch & Latch Edges

- Put high fan-out signals on globals
  - Reduces delays
  - Subject to resource availability
  - Global insertion delay may make this option not valid

$\}$ Changes T_{data}

- Turn on physical synthesis
  - Duplicates logic to reduce fan-out
  - Longer compilation time & higher utilization

$\}$ Changes T_{data}

Case 2) Possible Solutions (cont.)

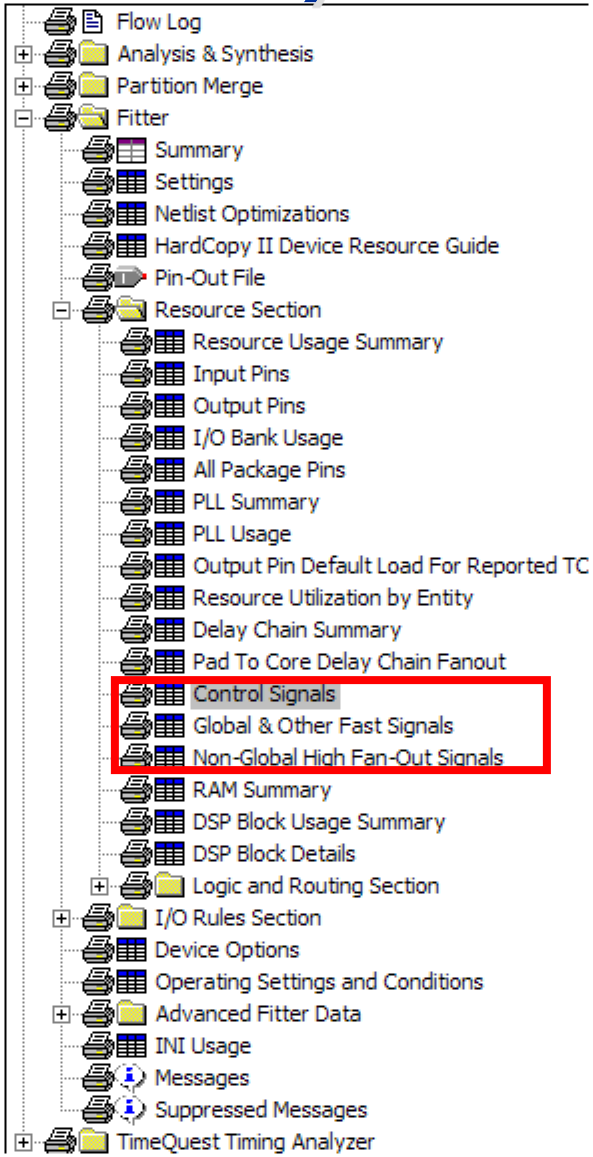
- Use max_fanout constraints
  - Simple to do
  - Trial & error process, multiple compiles

$\left.\vphantom{\rule{0pt}{4em}}\right\}$ Changes T_{data}

- Manual duplication of logic
  - Reduces fan-out
  - Allows user to intelligently control how each copy is used in design
  - May be a time intensive process depending on how signal is distributed

$\left.\vphantom{\rule{0pt}{6em}}\right\}$ Changes T_{data}

Case 2) Global Signals


- Flow Log
- Analysis & Synthesis
- Partition Merge
- Fitter
  - Summary
  - Settings
  - Netlist Optimizations
  - HardCopy II Device Resource Guide
  - Pin-Out File
  - Resource Section
    - Resource Usage Summary
    - Input Pins
    - Output Pins
    - I/O Bank Usage
    - All Package Pins
    - PLL Summary
    - PLL Usage
    - Output Pin Default Load For Reported TC
    - Resource Utilization by Entity
    - Delay Chain Summary
    - Pad To Core Delay Chain Fanout
    - Control Signals
    - Global & Other Fast Signals
    - Non-Global High Fan-Out Signals
    - RAM Summary
    - DSP Block Usage Summary
    - DSP Block Details
    - Logic and Routing Section
  - I/O Rules Section
  - Device Options
  - Operating Settings and Conditions
  - Advanced Fitter Data
  - INI Usage
  - Messages
  - Suppressed Messages
- TimeQuest Timing Analyzer

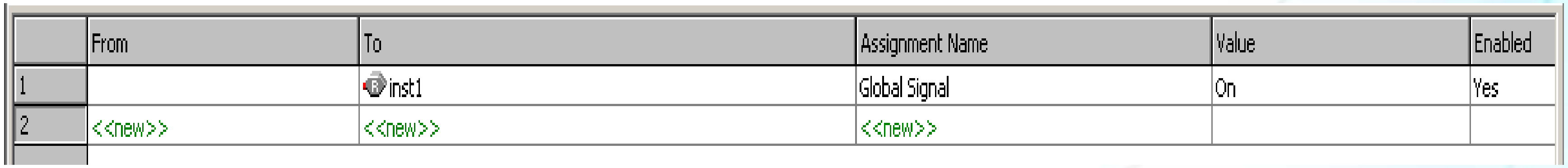
- Examine Fitter report for global & non-global signals
- Fixed number of global signals in a given device
- Fitter algorithms may auto-promote high fan-out signals (see fitter messages)

Case 2) Global Signals

- Manually promote signals with global assignment
- Thru TCL interface

```
set_instance_assignment -name GLOBAL_SIGNAL ON -to inst1
```

- Thru GUI



	From	To	Assignment Name	Value	Enabled
1		inst1	Global Signal	On	Yes
2	<<new>>	<<new>>	<<new>>		

Case 2) Physical Synthesis

- Options to try
  - Combinational physical synthesis
    - Performs duplication for combinatorial nodes
  - Register duplication

- See Quartus II handbook chapter “Netlist Optimizations & Physical Synthesis”
  - Explains features in detail
  - Lists caveats and exceptions

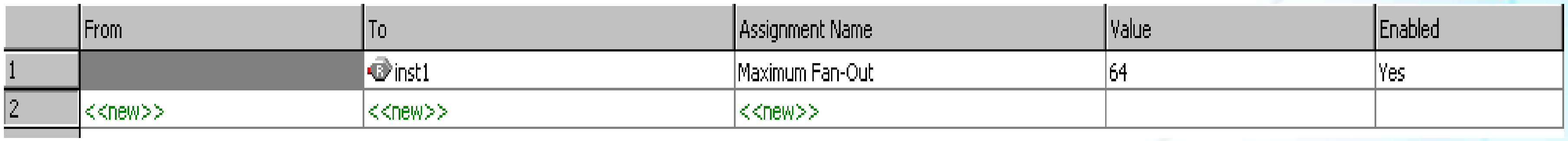
Case 2) MAX_FANOUT Constraint

- Controls the number of destinations so the fan-out count does not exceed the value specified

- Thru TCL interface

```
set_instance_assignment -name MAX_FANOUT  <integer> -to <instance>
```

- Thru GUI

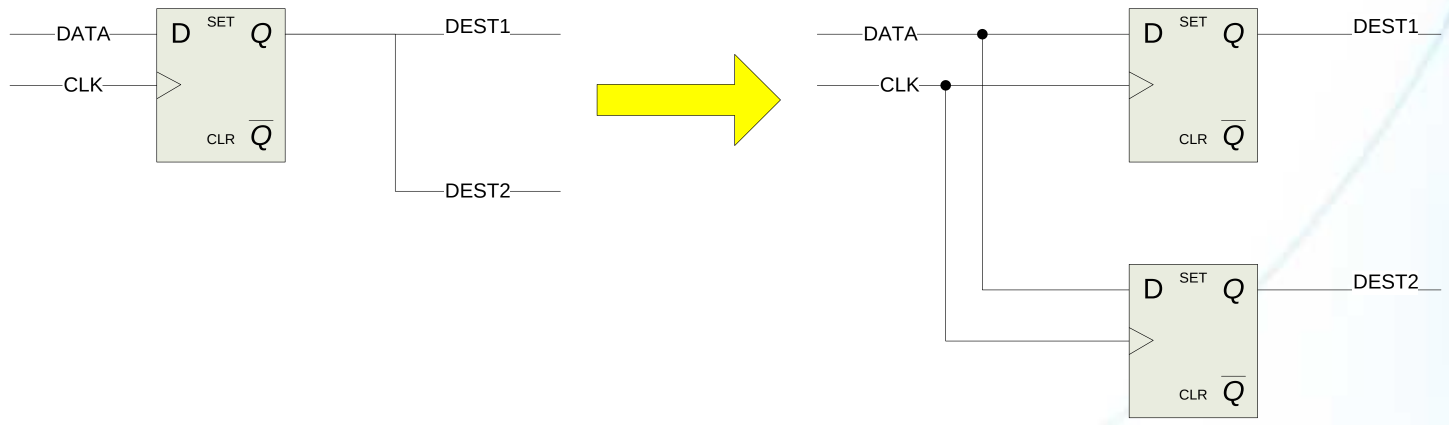


	From	To	Assignment Name	Value	Enabled
1		inst1	Maximum Fan-Out	64	Yes
2	<<new>>	<<new>>	<<new>>		

Case 2) Manual Duplication

- Two methods:
  1. Manual duplication in source code
  2. Manual Logic Duplication assignment

- Manual Logic Duplication Assignment
  – Duplicates the source node, and uses the new duplicated node to fan out to the destination node



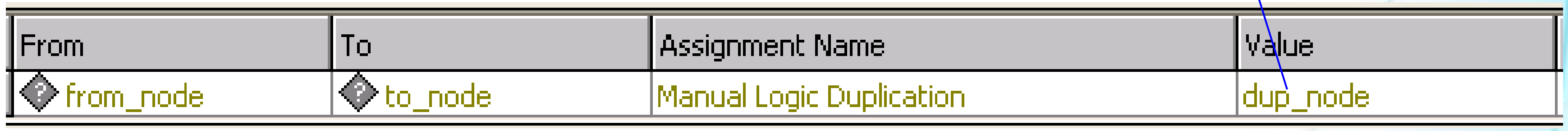
Case 2) Manual Duplication (cont.)

- Thru TCL interface

```
set_instance_assignment -name duplicate_atom dup_node -from from_node -to to_node
```

Name of duplicated node

- Thru GUI

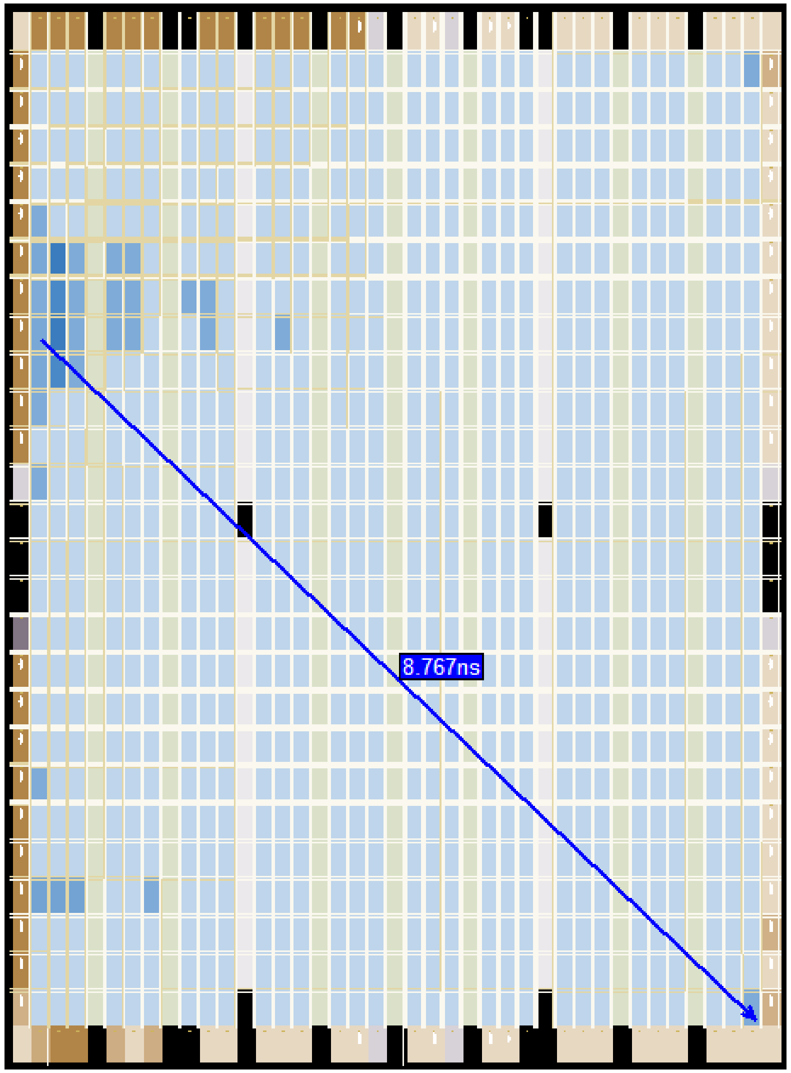


From	To	Assignment Name	Value
from_node	to_node	Manual Logic Duplication	dup_node

Case 3) Conflicting Physical Assignments

- Physical location assignments place registers too far apart
  - Increases T_{data} and setup analysis fails
- How to verify
  - Chip Planner
    - Locate timing path from TimeQuest

Case 3) Chip Planner



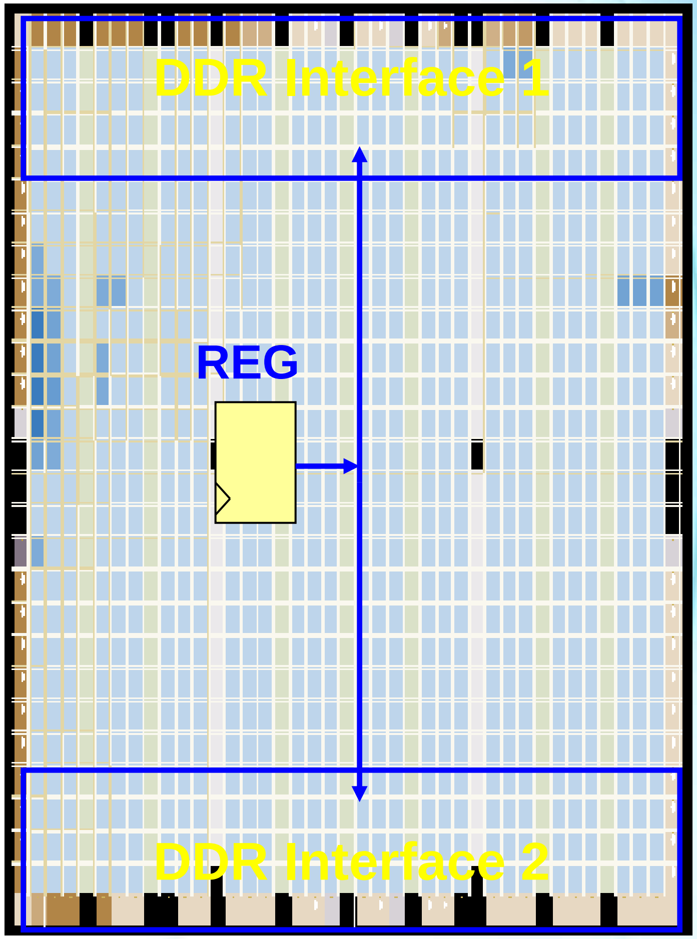
With setup issues, flops are usually too far apart

Why has fitter placed the flops so far apart?

Case 3) Explanation

- Due to conflicting Physical Requirements

- For Example
  - Memory interfaces on either ends of the device
  - Signals feeding both interfaces
  - No-Win scenario for Fitter
    - If REG is put near DDR I/F 1, path to DDR I/F 2 fails
    - If REG is put near DDR I/F 2, path to DDR I/F 1 fails



Case 3) Checks

- Which location constraints are interacting?
  - i.e. pin, register, etc.

- Are registers constrained to IO elements?
  - i.e. Fast {Input | Output | Output Enable} Register assignments

- Are there LogicLock Regions?

Case 3) Possible Solutions

- Add multi-cycle assignments if design allows

  } Changes Launch & Latch Edges

- Re-evaluate all location assignments
  - Simple to do
  - May be limited by design requirements

  } Changes T_{data}

- Turn on physical synthesis
  - Duplicates logic to reduce fan-out
  - Longer compilation & possibly higher utilization

  } Changes T_{data}

Case 3) Possible Solutions (cont.)

- Add pipeline registers
  - Reduces logic levels
  - Adds latency

$\}$ Changing T_{data}

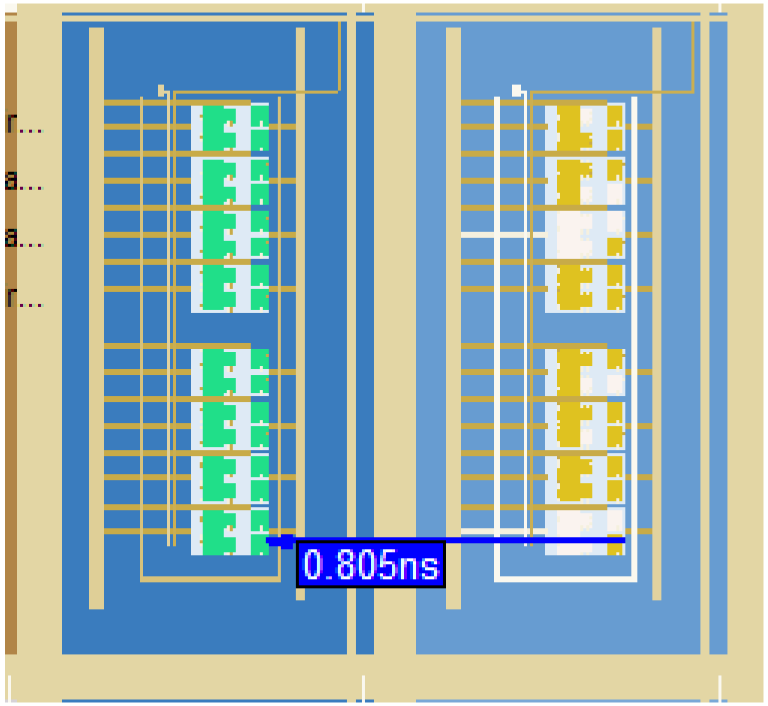
- Manual duplication of logic
  - Reduces fan-out
  - Long & laborious trial and error process

$\}$ Changing T_{data}

Case 4) Conflicting Timing Assignments

- Fitter can’t honor multiple assignments constraining path
  - Ex. Setup vs. hold; Clock vs. I/O
- How to verify
  - Use Chip Planner

Case 4) Chip Planner



With hold issues, flops are usually too close together

Why has fitter placed the flops so close together?

Case 4) Analysis

- Due to competing timing assignments

- Examining timing constraints that affect path
  - Examples
    - set_max_delay vs. set_min_delay
    - Path-based constraint vs. clock constraint

Case 4) Possible Solutions

- Add multi-cycle assignments if design allows

Changes Launch & Latch Edges

- Re-evaluate all timing assignments
  - Simple to do
  - May be limited by design requirements

Changes T_{data}

- Turn on physical synthesis
  - Duplicates logic to reduce fan-out
  - Longer compilation & possibly higher utilization

Changes T_{data}

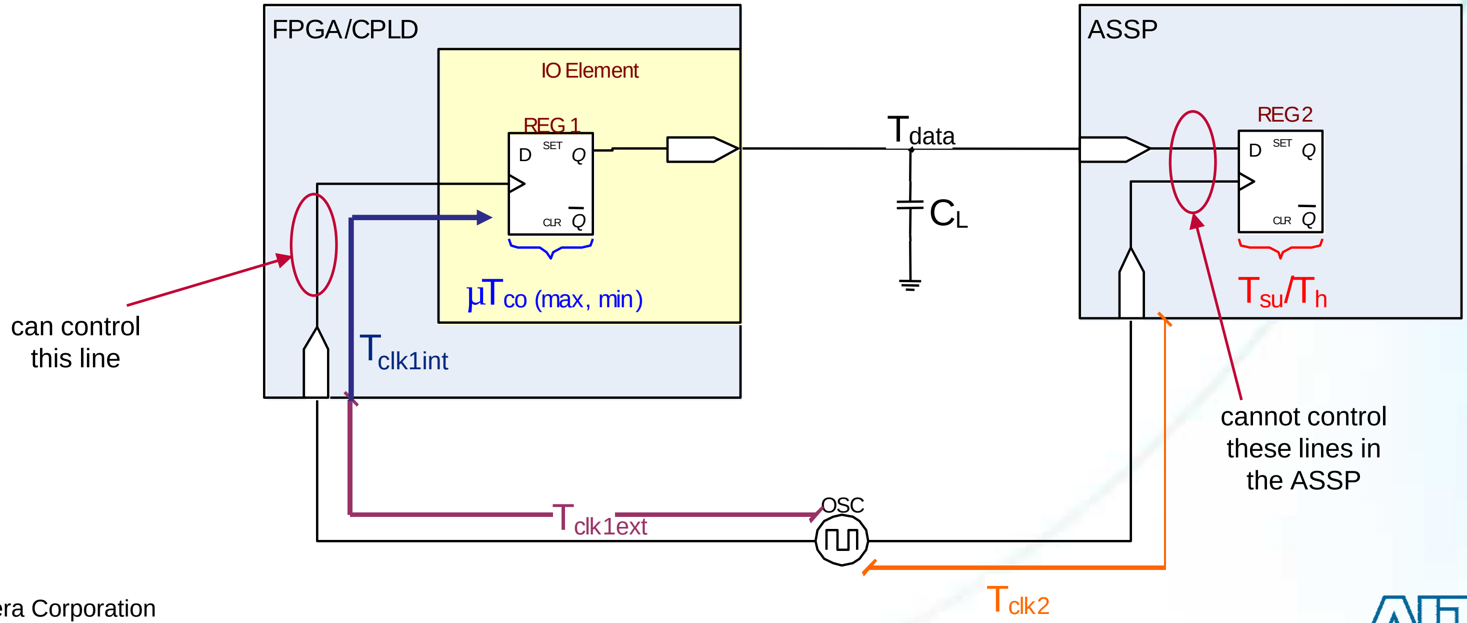
Case 5) Tight Timing Requirements

- Fitter can’t honor assignments as they are unobtainable

- How to verify
  - Use TimeQuest to verify path timing after all other cases have been ruled out
    - No fan-out, logic level, timing, skew or location issues

Case 5) Example: Output I/O Failing

- Flop is packed into IO element (best placement)
- Setup timing requirement is very tight
  - Board delays, capacitive loading, etc.
  - $T_{period} < (T_{co} + T_{data} + T_{CL} + T_{su})$
- How do you achieve timing?



Case 5) Slack Equations

Setup Slack Equation:

$$\underbrace{(\text{latch edge} + T_{clk2} - T_{su})}_{\text{Data Required}} - \underbrace{(\text{launch edge} + T_{clk1ext} + T_{clk1int} + T_{co} + T_{data})}_{\text{Data Arrival}}$$

What can we change?

Hold Slack Equation:

$$\underbrace{(\text{launch edge} + T_{clk1ext} + T_{clk1int} + T_{co} + T_{data})}_{\text{Data Arrival}} - \underbrace{(\text{latch edge} + T_{clk2} + T_{h})}_{\text{Data Required}}$$

Case 5) Slack Equations (cont.)

- Assuming that the board layout was done, we can make the following argument for change:

Setup Slack Equation:

$$\underbrace{(\text{latch edge} + \cancel{T_{clk2}} - \cancel{T_{su}})}_{\text{Data Required}} - \underbrace{(\text{launch edge} + \cancel{T_{clk1ext}} + T_{clk1int} + \cancel{T_{co}} + \cancel{T_{data}})}_{\text{Data Arrival}}$$

Hold Slack Equation:

$$\underbrace{(\text{launch edge} + \cancel{T_{clk1ext}} + T_{clk1int} + \cancel{T_{co}} + \cancel{T_{data}})}_{\text{Data Arrival}} - \underbrace{(\text{latch edge} + \cancel{T_{clk2}} + \cancel{T_{h}})}_{\text{Data Required}}$$

Case 5) What Can Be Changed?

- T_{su}, T_h, T_{co}, T_{data}, $T_{clk1ext}$, T_{clk2} are fixed values
  - Can't change these
- We can only change
  - Launch/latch edge relationship
  - Clock path delay inside the FPGA ($T_{clk1int}$)

Case 5) – Possible Solutions

- Add multi-cycle assignments if design allows

  } Changes Launch & Latch Edges

- Shift source clock
  - Pro: simple work-around
  - Con: subject to resource availability
    - If we shift source clock, we need to add multi-cycle assignment

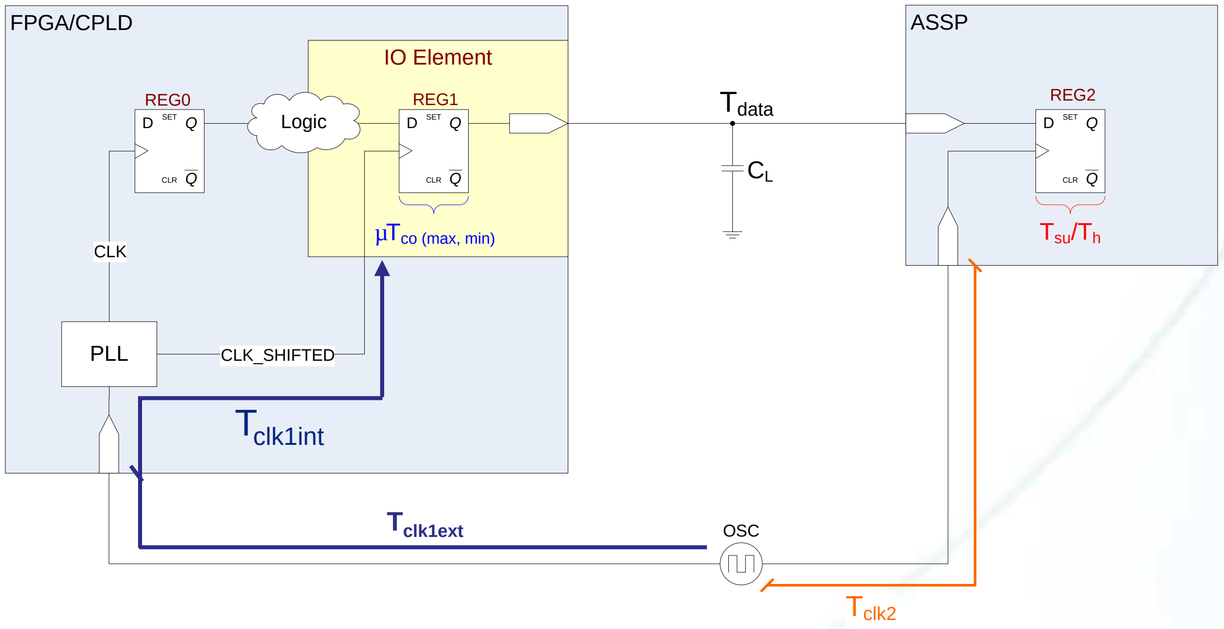
  } Changes Launch & Latch Edges

- Selecting faster clock routing, if available

  } Changes $T_{clk1int}$

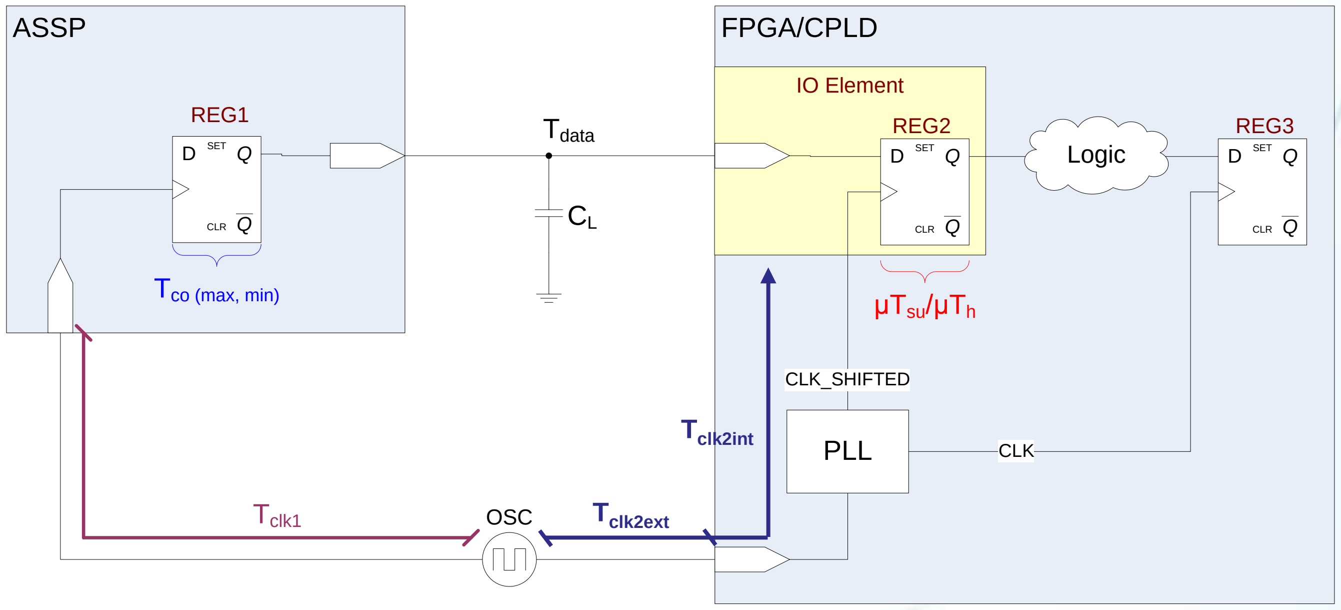
Case 5) Shifting $T_{clk1int}$

- Steal margin from REG0-REG1 timing to make up for shortfall in REG1-REG2 timing
- Use PLL to shift $T_{clk1int}$ by failing amount
  - Launch data earlier on REG1 with respect to input clock



Case 5) Input I/O Paths – shifting $T_{clk2int}$

- Handle similar to example output path
  - Re-evaluate input constraint
  - Use PLL to shift or delay clock ($T_{clk2int}$) driving input registers

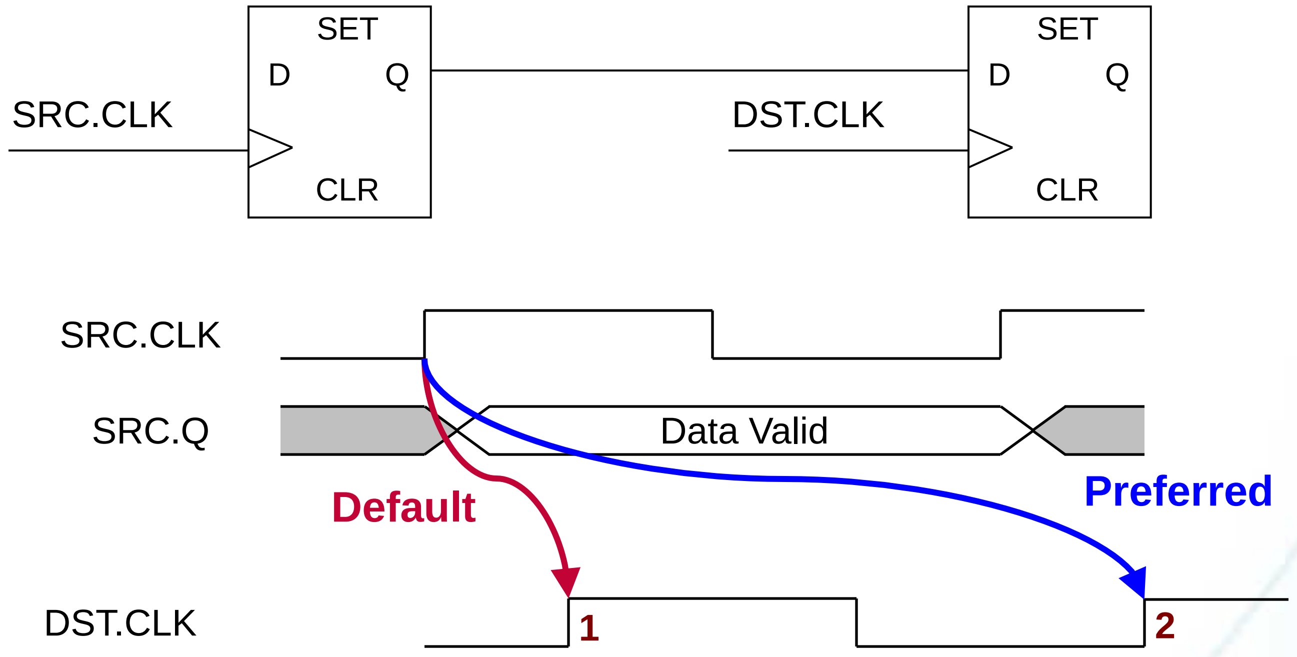


Case 5) Important Note on Shifting $T_{clk2int}$

- The destination clock is a delayed version of the source clock
- By shifting $T_{clk2int}$, a multi-cycle assignment is needed at the destination

Case 5) Example: Shifting $T_{clk2int}$

$F_{destination} = F_{source} + \text{Phase Shift}$



For this case: DMS = 2 and DMH = 1

Solving Timing Failures Review

1) Too many logic levels
2) Fan-out signals
3) Conflicting physical constraints
4) Conflicting timing assignments
5) Tight timing requirements

- These are common examples
- Must know and understand design to choose best solution