EE-382M

VLSI-II

Basic Timing Analysis for EDP

Matthew J. Amatangelo, Intel
Acronyms

• TA - Timing Analysis
• STA - Static Timing Analysis
• DCL - Delay Calculator Language
• AT - Arrival Time
• RAT - Required Arrival Time
• LCB - Local Clock Buffer
• EDP - Early Design Planning
• EDP-TC - Timing Closure for EDP
Early Design Planning for Timing Closure EDP-TC

• EDP-TC What is It?
• EDP-TC Goals & Objectives
• EDP-TC Starting Point Data Requirements
• EDP-TC Methodology How-To
• EDP-TC End Products
• Specifics for the Class Project: EDP-TC Floorplanning for Design Space Exploration & Timing Closure
EDP-TC What Is It?

- The process to identify and close on chip area and timing objectives and constraints during the microarchitectural design phase.
- Rapid Design space exploration during microarchitectural phase
  - Drive changes to the microarchitecture to enable achieving area and timing goals.
  - Enabling Rapid Convergence on Area & timing closure during design implementation phase.
EDP-TC Goals & Objectives

- **End result** is a microarchitectural starting point that is known in advance to have an implementation that can meet the program goals for not just area but also timing
  - Provide a starting point for the initial chip floorplan and sub-block physical design with the constraints required to meet the various timing objectives
  - Identify early in the design process (during the microarchitectural design phase) key timing problems intrinsic to the fundamental architectural mechanisms in the design.
- Get designers thinking about physical implementation required to meet the various timing objectives while still in the microarchitectural design phase
- Give designers a methodology & process for:
  - rapidly evaluating the microarchitectural and timing effects of chip physical design decisions (rapid design space exploration).
  - chip floorplanning targeted at closing not just area but also all key timing requirements.
Nature of EDP-TC

- Simplified analysis compared to implementation phase
  - Using 1 pvt late mode timing point
    • Assume monotonic switching per gate (no MIS)
    • Some pessimism built into uncertainty
  - Parasitics are estimated and based on placement
    • During implementation phase the goal will be to use extracted parasitics
  - Wires between blocks assume some max edge rate
    • i.e., virtual repeaters, time of flight wire delay calculations
  - Transparency is not modelled
  - All arrival and required times are absolute (class project)
    • All launch/capture pairs assumed synchronous
    • Analysis performed without LCBs
EDP-TC Starting Point Data Requirements

- Initial chip size, form factor and I/O requirements.
- Initial chip timing goals.
- Initial top level floorplannable block list & functionality.
- Initial chip & top level floorplannable block connectivity.
- For each floorplannable block
  - initial sizes
  - initial form factors
  - initial pin positions
  - initial timing assertions
- These initial starting points normally evolve during the EDP-TC process.
EDP-TC Methodology How-To

- Methodology Overview
- Block Size Estimation (another lecture)
- Block Timing Assertions Generation
  - How do you get the numbers
- Delay Estimation
Methodology Overview (Big Picture)

- Determine chip I/O definition from architectural specification
  - I/O placement (next levels of packaging & system considerations)
- Determine initial cut at top level floorplannable blocks from architectural and/or functional descriptions and specifications.
- Generate first pass top level netlist specifying interconnection of top level floorplannable blocks and chip I/O’s
- Estimate initial top level floorplannable block sizes
  - Analyze the block’s component parts
    - Use prior implementations of similar functions as a starting point
    - Perform first pass logic realization on some sub-blocks
- Estimate chip size
  - Floorplannable block area + wiring uplift (~30%)
Methodology Overview (Big Picture - cont)

- Produce chip floorplans
  - determine initial form factors
    - block attributes (memory cell)
    - connectivity (bus widths)
    - wireability
- Iterate on floorplan to close area & timing constraints
  - Given initial floorplan, estimate timing of top level critical timing paths based on top level connectivity, block placement, and pin placements
  - Modify block form factor, placement, pin placement and architectural/functional description if required to improve timing and or area.
    - Changes to architectural specifications will yield updates to the number of blocks, their sizes and/or form factors, and the netlist (connectivity) of the top level blocks.
- Done when you have an architectural specification and a floorplan that achieves area and timing goals.
Block Timing Assertions Generation

• Block Timing Assertions - What Are They?
• Usage of Block Timing Assertions in EDP-TC.
• Clock Cycle Adjusts in Slack Calculations.
• Estimating Delays for Initial Floorplans.
• How Timing Contracts (Block Assertions) Are used in the Implementation Phase of the Design.
Block Timing Assertions --- What Are They?

• Basic Block Timing Model
  – Depicts timing information about paths in a particular block
    • 3 types of paths modeled in a block
      – capture: block input to register
      – launch: register to block output
      – purely combinatorial: delay from block input to output

• Basic Block Assertions
  – Input Pin Required Arrival Times (RAT)
    • For each input pin on a block
      – latest time a signal can arrive at that pin and still get successfully captured in the register inside the block fed by that pin.
        » Calculated by: \( \text{RAT} = \{\text{AT(clock @ register)}\} - \{\text{Internal logic & wire delay between pin and register}\} - \{\text{register setup requirement}\} \)
    – combinatorial: \( \text{RAT} = \) Need to analyze entire path from register launch to register capture, along with combinatorial delay for the portion of the path inside this block.
Block Timing Assertions --- What Are They? (con’t.)

• Basic Block Assertions (con’t).
  – Output Pin Arrival Times: (AT)
    • For each input pin on a block
      – latest time that a signal launched from a register inside the block
        that feeds the pin arrives at the pin.
        » Calculated by: AT = \{AT(clock@register)\} + \{Internal logic &
          wire delay between register and pin\} + \{register launch delay\}
      – combinational: AT = same problem as combinational RAT described
        on preceding page.
  – Block assertions determined by block alone except for purely
    combinational paths
    • Preferable to eliminate if possible both wire feed-throughs &
      purely combinational paths from all top level blocks.
      – Want assertions & block timing properties to be floorplan
        independent to enable rapid iteration.
Path Types Modelled in a Block

RAT: determined by clk arrival

RAT = AT(clk) - Din - Setup

RAT: determined by capture register block & global wire delay & Dinout

Internal logic & wire delay from input pin to register

AT: determined by launching register block & global wire delay & Dout

AT = AT(clk) + Dout + Dregister

Internal logic & wire delay from register to output pin

Delay for wire and combinatorial logic

AT: determined by launching register block & global wire delay & Dinout
Usage of Block Timing Assertions in EDP-TC

- Every pin of every block and the chip top level block has both an AT and a RAT.
  - Connectivity determines which are combined to determine the slack (timing goodness) of a path.
- Calculate the slack for a path sourced from one block and sunk in another.
  - Avoid purely combinatorial paths and feedthroughs when possible
    - Avoid these at the full chip level
  - Slack calculation must consider phase of launching and capturing clocks in a path
    - all events derived from one cycle of the master clock (ignore multicycle paths for now)
    - no zero cycle setup paths exist
    - A cycle adjustment is made to this calculation when the leading edge of the master clock corresponds to the capture event of the path and the trailing edge corresponds to the launching event.
- When all paths have slack \( \geq 0 \) the block assertions constitute the Timing Contracts for each block.
Assertion Generation for Purely Combinatorial Paths

RAT: determined from capturing block RAT' - Dwire2 - Dinout

AT: determined from clk launch source block AT' + Dwire1 + Dinout
Usage of Block Timing Assertions

Slack(path of X.clk->Y.pin) = RAT(Y.pin) - \{ AT(X.pin) + Dwire \} + Adjust
**Path Slack Calculation Adjust**

- **Launch Edge** | **Capture Edge** | **Adjust** *
- leading | leading | m cycles
- trailing | trailing | m cycles
- leading | trailing | m-1 cycles
- trailing | leading | m cycles

* Assuming m cycle path, e.g., typical single cycle paths m=1; two cycle path m=2

For class project, include Adjust in RAT:

\[
\text{RAT} = \text{AT}(\text{clk}) - \text{Din} - \text{Setup} + \text{Adjust}
\]

\[
\text{Slack(path of X.clk->Y.pin)} = \text{RAT}(\text{Y.pin}) - \{ \text{AT(X.pin)} + \text{Dwire} \} + \text{Adjust}
\]
How Timing Contracts are Used in the Implementation Phase of Design

- Implementation phase starts at the end of EDP-TC.
- Given that EDP-TC closed chip timing at 0 slack, the Block Assertions are the Timing Contracts.
- Each block during design is timed stand alone against these contracts, or budgets. Affects synthesis (auto or manual).
  - The RATs are now the assumed arrival times at the blocks inputs.
  - The ATs are now the assumed required times at the blocks outputs.
- The contracts (assertions) are typically periodically updated from full chip timing runs to reflect actual design changes.
  - It’s important to continue to have a complete & consistent set of contracts that, if achieved by each block, yields a chip which meets the timing objective.
E.g., Contracts applied to block level timing

From Full Chip Level Timing:

Block X
Level Timing: RAT(X.pin) = 600

Block Y
Level Timing: AT(Y.pin) = T-100

Let Slack(path of X.clk->Y.pin) => 0

Dwire = 200

AT(X.pin)

RAT(Y.pin)
Wire Delay Estimation

- Wire delay calculation & analysis overview.
- Elmore Delay
- Wire Delay Estimation Summary
  - Time of Flight
  - Elmore Delay
Analyzing On-Chip Interconnect

- **Simplified interconnect analysis.**
  - **Time of Flight (EDP-TC)**
    - Simplest approach for EDP-TC.
      - Given in picoseconds per millimeter
      - Assume optimal signal regeneration (buffering satisfies max allowable slew)
    - Routing parasitic expressed as some delay per unit distance
    - Determined for the process technology with spice simulations
    - Assume certain levels of interconnect (parallel plate and fringing fields), coupling, and buffering
  - **Lumped RC product**
    - Overly conservative for long wires.
  - **RC Ladders.**
    - Limiting Case, \( R \times C \times (\text{Length}^2 / 2) \).
  - **Elmore Delay Model.**
    - Typically much less conservative from RC Ladders.
    - Effective estimates for Multi-Drop Nets.
- **Save more complex analysis for implementation phase**
  - Shielding, inductance, 3D fields, etc. (poles/residues, AWE, 3D field solvers, …)
Elmore RC Delay Calculation Model

- More realistic RC delay than lumped RC for long nets.
- Able to handle multi-drop nets.
- The formula can be written from inspection of the RC tree.
- Calculable in linear time.
- Provable upper bound on RC delay.
  - Can still significantly overestimate RC delay in some cases.
Elmore RC Delay Calculation Model (con’t).

\[
Td_6 = R_1 C_1 + (R_1 + R_2)(C_2 + C_7 + C_8) + \left( \sum_{n=1}^{3} R_n \right) (C_3) + \left( \sum_{n=1}^{4} R_n \right) (C_4) + \left( \sum_{n=1}^{5} R_n \right) (C_5) + \left( \sum_{n=1}^{6} R_n \right) (C_6)
\]
Wire Delay Estimation Summary

• Time of flight is simplest and probably best for initial floorplan timing.
  – Use delay per wire length that considers best estimate for technology, routing layers, coupling, etc. as measured in early circuit analysis (spice)

• Use Elmore Delay on selected nets as more estimated routing information becomes available
  – Especially if the use of wide wires or upper level metal for low impedance wiring is required to close timing
EDP-TC End Products

• What comes out of the process
  – floorplan
  – block size & shape (discussed in another lecture)
  – pin positions
  – timing contracts (assertions)
Specifics for the Class Project
EDP-TC Floorplanning for Timing Closure

• Starting point is Verilog description
  – Embodies architectural specification
  – Chip I/O boundary is given
  – Top level floorplannable blocks and connectivity specified in Verilog

• First Step - Estimate Block Sizes & Shapes

• Step 2 - Determine Chip size & shape & initial I/O placement based on step 1, the Verilog, and class input assumptions.
  – In this class we are only concerned with chip size.

• Create initial placement based on size, shape & connectivity
• Create initial timing assertions for each block based on functionality
• Iterate on chip floorplan, block placement, pin placement, routing, engineering wires and block definition & assertions utilizing information derived from the timing process.
Initial Floorplan Timing Activities for Class Project

Block owner estimates block size and delivers to integrator

Chip Floorplan

Run Timing Script

Slack >= 0?

NO

YES

Floorplan file

Floorplan file

Block owner creates/updates AT/RAT estimates to assertion file

Re-Floorplan

Floorplan file

constraints

ATs/RATs == Contracts
Prior Results Based Constraints

Slack(path of X.clk->Y.pin) = RAT(Y.pin) - { AT(X.pin) + Dwire } + Adjust
If pass-thrus not significant, Let: Budget = Tcycle –Dwire;
and let the reduced delay:  r = Budget/[AT(X.pin)+RAT(Y.pin)];
Otherwise, Let Budget = Tcycle and Let r = Tcycle/(Tcycle – Slack)

For Block X resynthesis, if slack < 0, set new AT(X.pin) to
r [ AT(X.pin)]
..and set new RAT(Y.pin) to
r [ RAT(Y.pin)]
Create Initial Timing Assertions

- Verilog describes block’s functional definition
- Cycle time: ? GHz
- I/O timing specification: Define it if one doesn’t exist
- Assume \( t = 0 \) at the node where clock is delivered to the blocks sequential logic
  - This is an EDP approach to avoid concern with estimating the delay through the LCB (local clock buffer). Skew associated with LCBs will be handled by timing tool.
- Estimate register launch delay and required setup time
- Assume 0ps of delay for clock skew, jitter and mistracking, normally accounted for in timing analysis tool. Need to insert explicitly if analysis is done “by hand”.
Create Initial Timing Assertions (cont’d)

- **Estimate block component implementation**
  - Derive propagation delays between INPUTS, OUTPUTS, and registers (prior examples)
  - Turn these delays into block ATs and RATs
- **Note:** Internal (reg-to-reg) paths must meet frequency targets. In other words, these AT/RAT files assume internal paths of the cluster meet frequency.
  - Essentially representing clusters with blackbox model
    - Internal paths hidden
    - Models are inherently appropriate up to the max frequency of that designed (and duty cycle, if half cycle paths used)
  - Run at lower level of hierarchy to include internal paths
Perform Timing Analysis

- Calculate RATs for block inputs, ATs for block outputs
  - consider initial timing model including clock arrival
    - establish block-consistent clock arrival at registers
- Iteration to zero slack
  - can change:
    - wire delay == floorplan and/or block pin positions (if process supports this)
    - assertions
      - launch time (block design)
      - capture time (block design)
      - arc delays
Capture Results

- Floorplan
- Timing contracts
- Pin positions
How to run the in-class timer

Block pin names and location (.v)
Top level connectivity (.v)
Wire metal

Netlist

block timing model (atrat)

timer

RESULTS
Synopsis of the ATRAT files:

<table>
<thead>
<tr>
<th>ARCS</th>
<th>TESTS</th>
<th>ATs</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1: x-&gt;t, delay=100</td>
<td>b5.x, RAT=800</td>
<td>io.pi1, 200</td>
</tr>
<tr>
<td>b2: -</td>
<td>b1.y, 900</td>
<td>io.pi2, 200</td>
</tr>
<tr>
<td>b3: x-&gt;t, 200</td>
<td>io.pi1, 1000</td>
<td>b1.t, 400</td>
</tr>
<tr>
<td>b4: x-&gt;t, 100</td>
<td>io.pi2, 1000</td>
<td>b2.t, 300</td>
</tr>
<tr>
<td>b4: x-&gt;u, 200</td>
<td>b6.x, 800</td>
<td>b6.t, 600</td>
</tr>
<tr>
<td>b4: y-&gt;t, 300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b4: y-&gt;u, 400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b5: x-&gt;t, 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b5: y-&gt;t, 200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b6: -</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

clk

0 ? 1000
Timer Files

- ATRAT files from block owners:
  - define (1) signal arrival times, (2) required arrival times, or tests, and (3) delays through circuit elements. Every element type in the design must have an ATRAT file; in this case: b1.atrat, b2.atrat, b3.atrat, b4.atrat, b5.atrat, b6.atrat, io.atrat

- Netlist file:
  - delineates the element connectivity including pin type, location, and wire type. The netlist file in this test case is ad.layout. For the top level view of the class project this file is owned by the integration team.

- Output files:
  - log - execution run log
  - pathlog - details of paths encountered
  - slackrpt - succinct slack-ordered path report
Input File Example: atrat file

BLOCK_NAME b1
START_AT_SECTION
PIN t clk rise 400
START_RAT_SECTION
PIN y clk rise 900
START_PASS_THROUGH_SECTION
PASS_THROUGH x t 100

output pins, launching event and time from launch that signal arrives at pin

input pins, capture event and time after (for now) this event that the arrival is required

in-to-out arcs, paths from input pins to output pins with propagation delay
Netlist Example: ad.layout

pin name and location relative to block origin point

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>pin</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_b1</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b2</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b3</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b4</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b5</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b6</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>i_b3a</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>io</td>
<td>I</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>t</td>
<td></td>
</tr>
</tbody>
</table>

The University of Texas at Austin
Output

Log messages:

### WARNING: Net wi21 has multiple drivers

### ERROR: Cannot propagate beyond pin(i_b2:x) -> Missing RAT!

### WARNING: Found a loop - breaking loop at i_b4:b4:u

Backtracking loop traversal

\[
i_b3:b3:x \leftrightarrow i_b5:b5:t \leftrightarrow i_b5:b5:y \leftrightarrow i_b4:b4:u \leftrightarrow i_b4:b4:x \leftrightarrow i_b3:b3:t \leftrightarrow i_b3:b3:x \leftrightarrow i_b5:b5:t \leftrightarrow i_b5:b5:y \leftrightarrow i_b4:b4:u
\]

Paths:

\[
i_b6:t \rightarrow i_b4:y \rightarrow i_b4:u \rightarrow i_b5:y \rightarrow i_b5:t \rightarrow i_b3:x \rightarrow i_b3:t \rightarrow i_b4:x \rightarrow i_b4:u \rightarrow i_io:po2 -816.629
\]

\[
i_b6:t \rightarrow i_b4:y \rightarrow i_b4:u \rightarrow i_b5:y \rightarrow i_b5:t \rightarrow i_b3:x \rightarrow i_b3:t \rightarrow i_b4:x \rightarrow i_b4:t \rightarrow i_io:po1 -707.747
\]

\[
i_b6:t \rightarrow i_b4:y \rightarrow i_b4:u \rightarrow i io:po2 -73.55
\]

\[
i_b6:t \rightarrow i_b3a:x \rightarrow i_b3a:t \rightarrow i_b1:y -66.1875
\]

\[
i_b1:t \rightarrow i_b5:x \rightarrow i_b5:t \rightarrow i_b3:x \rightarrow i_b3:t \rightarrow i_b4:x \rightarrow i_b4:u \rightarrow i io:po2 -64.3805
\]

\[
i io:pi1 \rightarrow i_b1:x \rightarrow i_b1:t \rightarrow i_b5:x \rightarrow i_b5:t \rightarrow i_b3:x \rightarrow i_b3:t \rightarrow i_b4:x \rightarrow i_b4:u \rightarrow i io:po2 15.845
\]
Back up slides
Delay Estimation Example for Initial Floorplanning using Latches

• Guidelines
  – For illustration, times are taken from initial Power 4 design.
    • These times should be adjusted accordingly for our semester project parameters.
  – Assume $t = 0$ at the clock mesh, and similarly at the input to the local clock buffers (LCBs).
  – Assume 2 FO4 for propagation delay through the LCB to the clock launch or capture logic.
    • For simplicity, you can skip this step and assume the clock arrives at the latches at $t = 0$ as long as delayed clocks are not used.
  – Assume 1 FO4 of latch launch delay, and another FO4 for latch setup and capture delay.
  – Assume 0ps of delay for clock skew, jitter and mistracking, normally accounted for in timing analysis tool. Need to insert explicitly if analysis is done “by hand”.
    • In Power 4 initially assumed to be 100ps, later reduced to 50ps.
      – Hardware measurements indicate 35 ps.
Initial Delay Estimation (con’t).

- Assume 1 FO4 of delay per logic stage, or estimate logic delay of the path in question and round up to the nearest FO4 ps.
- Assume 1 FO4 of delay for the RC of the global interconnect, estimated from a Steiner routing of the net. Usually supplied by the global integrator from the initial floorplan.
  - Many early timing / floorplanning systems will generate this data automatically from the floorplan.
- Initially assume all block pins are in the middle of the entity.
  - Place pins based on initial floorplan and timing.
  - Refine based on wireability and timing.
- Assign clock phase information to block pins based on the last clock phase to control the signal prior to it reaching the pin.
  - That is, do not trace back through transparent latches.
Departure Time Estimation (ATs)

Data

Static L1 Latch, No Cycle Stealing

Dynamic Logic, Phase 2

Dynamic Logic, Phase 1

C1 Leading Edge Reference Time

C2 Leading Edge Reference Time

Static Transparent L2 Latch with Cycle Stealing

Data Launched

Clock arrival reflects Launch event
Required Arrival Time Estimation (RATs)

Clock arrival reflects Capture event
Delay Estimation - Early Floorplanning

Waveforms at Clock Mesh

RC Delay from Global Integration

Entity A

Entity B

Input Pin Capacitance

Arrival Time

Departure Time

Drive Strength

Clock Mesh

LCB

Logic & Wire

Latch L2

Logic & Wire

Phase L2

t = 0

200ps

C2

x ps

t = x + 200

100ps

t = x + 300

y ps

eta assertion

Entity A

M Clock

0 500 1000

C1 Clock

C2 Clock

Phase L2

Latch L1

200ps

200ps

200ps

C1

LCB

t = 1200

200ps

t = 1000

Input Pin Capacitance

PIS assertion

t = 1100 - z

100ps

t = 1200 - 100 = 1100

EE 382M Class Notes
Block Assertion Example Summary

<table>
<thead>
<tr>
<th>Pin</th>
<th>AT</th>
<th>Pin</th>
<th>RAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.q0</td>
<td>1000</td>
<td>B.in0</td>
<td>-200</td>
</tr>
<tr>
<td>A.q1</td>
<td>900</td>
<td>B.in1</td>
<td>300</td>
</tr>
<tr>
<td>A.q2</td>
<td>700</td>
<td>B.in2</td>
<td>-500</td>
</tr>
<tr>
<td>A.q3</td>
<td>500</td>
<td>B.in3</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path</th>
<th>Type  *</th>
<th>Slack  *</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.q0 -&gt; B.in0</td>
<td>TL-TC</td>
<td>-200 -1000 -w + cyc</td>
</tr>
<tr>
<td>A.q1 -&gt; B.in1</td>
<td>TL-TC</td>
<td>300 -900 -w + cyc</td>
</tr>
<tr>
<td>A.q2 -&gt; B.in2</td>
<td>LL-LC</td>
<td>-500 -700 -w + cyc</td>
</tr>
<tr>
<td>A.q3 -&gt; B.in3</td>
<td>LL-LC</td>
<td>100 - 500 -w + cyc</td>
</tr>
</tbody>
</table>

* Notes:
  
  TL - trailing edge launched
  TC - trailing edge capture
  LL - leading edge launched
  LC - leading edge capture
  w = corresponding wire delay
  cyc = cycle adjust (1000ps in this example)