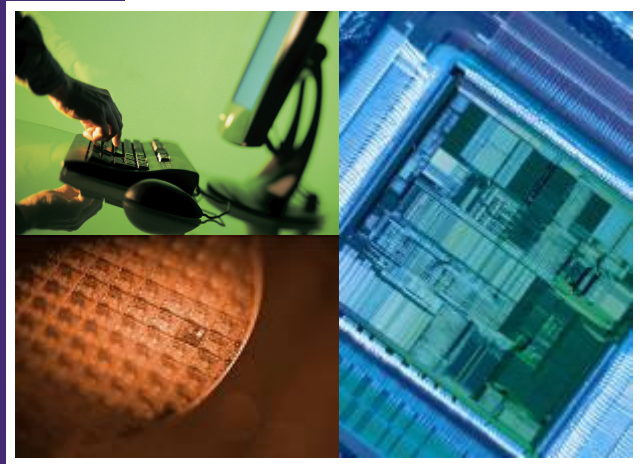
JupiterXT Timing Budgeting



Wei-Chun Chou
weichun@synopsys.com

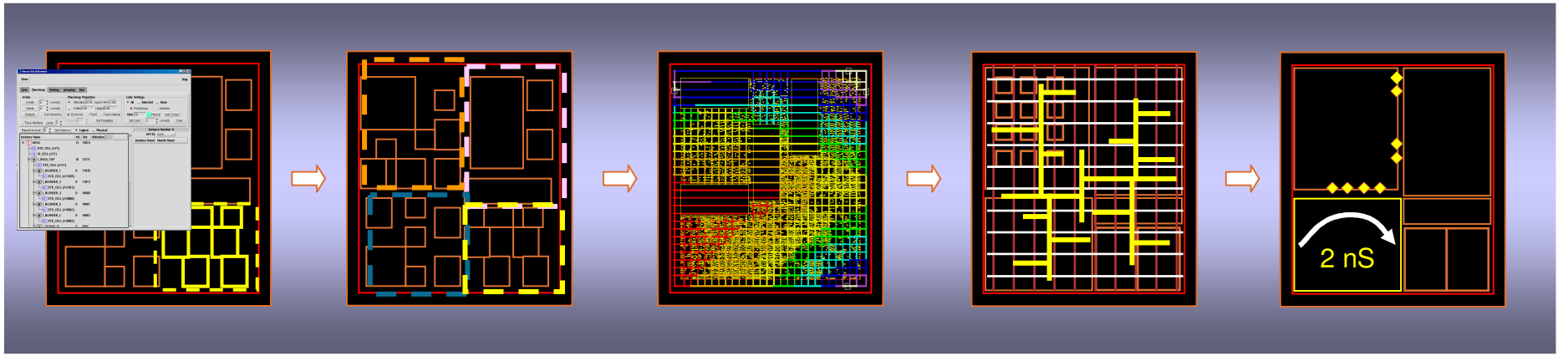
4-4-2007

R&D: Vasiliki Chatzi, Vineet Gupta
Moderator: Sandy Hsu

SYNOPSYS®
Predictable Success

Wei-Chun Chou

JupiterXT™ Hierarchical Flow Overview



Logical Partitioning, Plangroup Creation

Simultaneous Standard Cell Hard Macro Placement

Power Network Synthesis & Analysis

IPO Clock Planning Global Route

Route-Based Pin Assignment, Budgeting, and Commit Soft Macro

Plangroup: physical representation for logic hierarchy

Main objective: create good floorplan with pin locations and timing constraints for block implementation

Outline

- **Budgeting overview**
- Pre-budgeting flow
- Budgeting flow

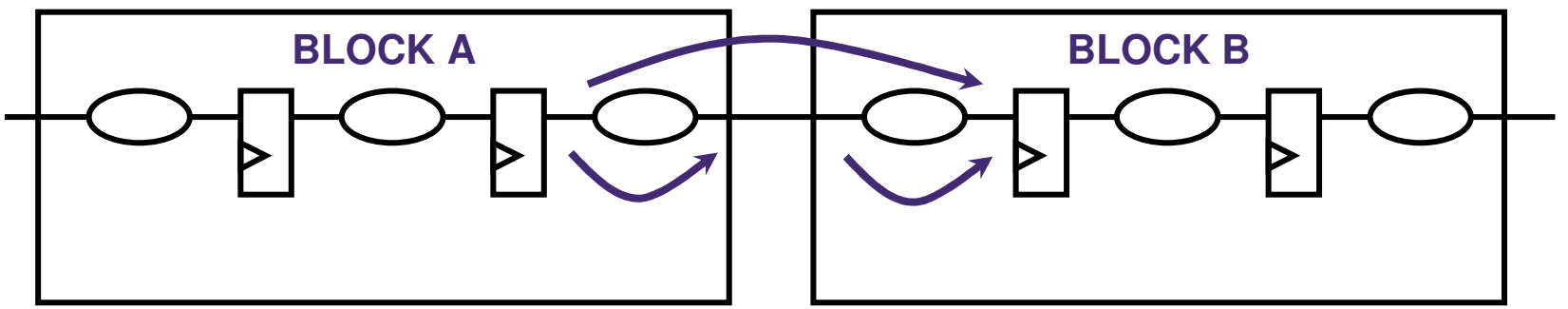
Why Budgeting?

- In hierarchical design methodology, budgeting generates SDC timing constraints for block-level implementation
  - Good budgeting inputs generates SDC with good quality
    - No over-constrained, or under-constrained SDC
  - Good SDC achieves good implementation of blocks
    - **Early detection of feasibility of top-level timing closure**

What is Budgeting?

The process of distributing positive and negative slack and creating block-level budget SDC files

- Budgeter determines input and output delays by analyzing delays of interblock timing arcs



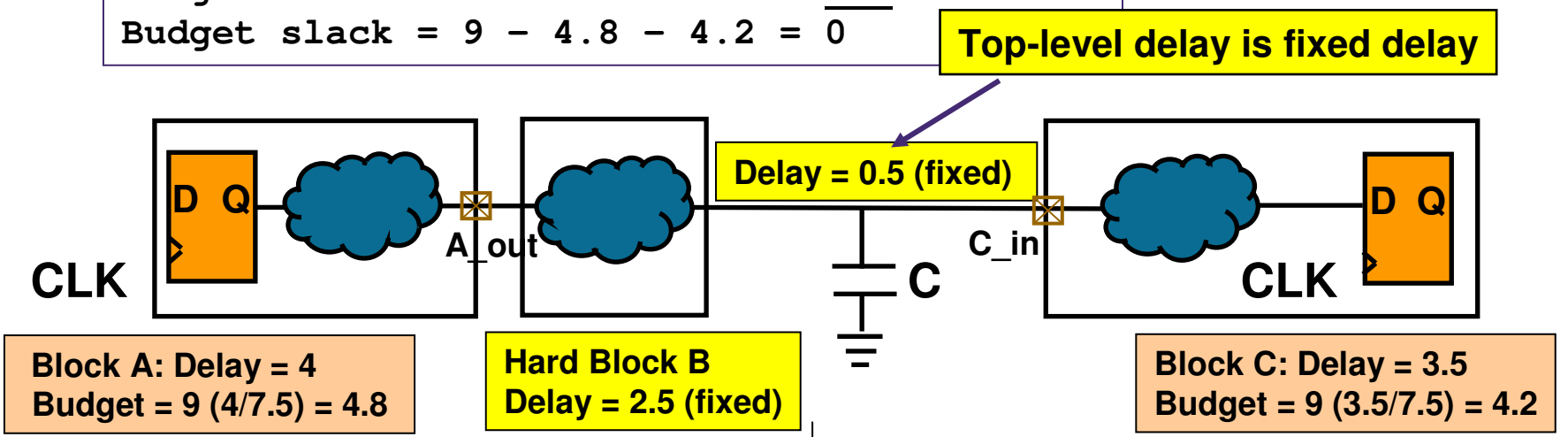
- Budgeter does not modify timing, only distributes slacks among blocks: **budget allocation**

Wei-Chun Chou

Proportional Budget Allocation Positive Slack

Budget = Total Budgetable Delay * Block Delay Percentage

```
CLK cycle time = 12
Total budgetable delay = 12 - 2.5 - 0.5 = 9
Original slack = 9 - 4 - 3.5 = 1.5
Budget slack = 9 - 4.8 - 4.2 = 0
```



Block A SDC file :

set_output_delay 7.2 –clock CLK [get_ports A_out]

output_delay = 2.5 + 0.5 + 4.2 = 7.2

Block C SDC file:

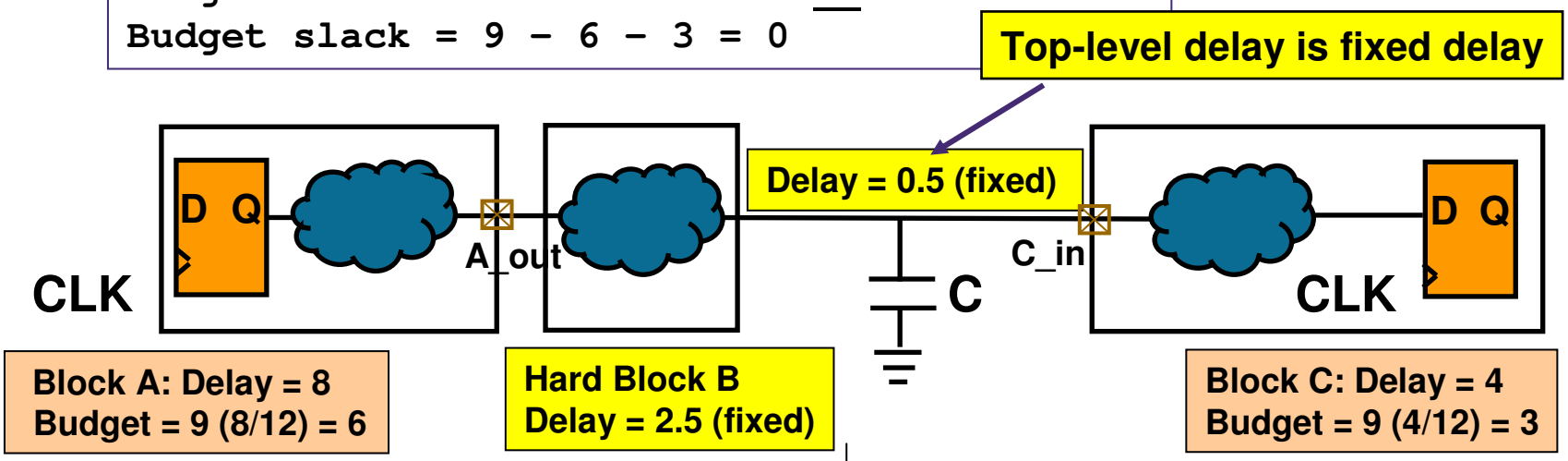
set_input_delay 7.8 –clock CLK [get_ports C_in]

input_delay = 4.8 + 2.5 + 0.5 = 7.8

Proportional Budget Allocation Negative Slack

Budget = Total Budgetable Delay * Block Delay Percentage

```
CLK cycle time = 12
Total budgetable delay = 12 - 2.5 - 0.5 = 9
Original slack = 9 - 8 - 4 = -3
Budget slack = 9 - 6 - 3 = 0
```



Block A SDC file :

set_output_delay 6 –clock CLK [get_ports A_out]

output_delay = 2.5 + 0.5 + 3 = 6

Block C SDC file:

set_input_delay 9 –clock CLK [get_ports C_in]

input_delay = 6 + 2.5 + 0.5 = 9

Budget Allocation Approaches

- Traditional approach:
  - Allocate budget proportionally to existing block delay
- Advanced approach:
  - Allocate budget proportionally to block delay based on virtual IPO result
    - Virtual IPO: only estimate IPO effect without actually changing netlist
- User-defined approach:
  - Specify percentage or fixed budget per block or per clock
- Budgets can be further tightened or relaxed in block implementation using a command:
  - set_context_margin
    - Give margin for early netlist

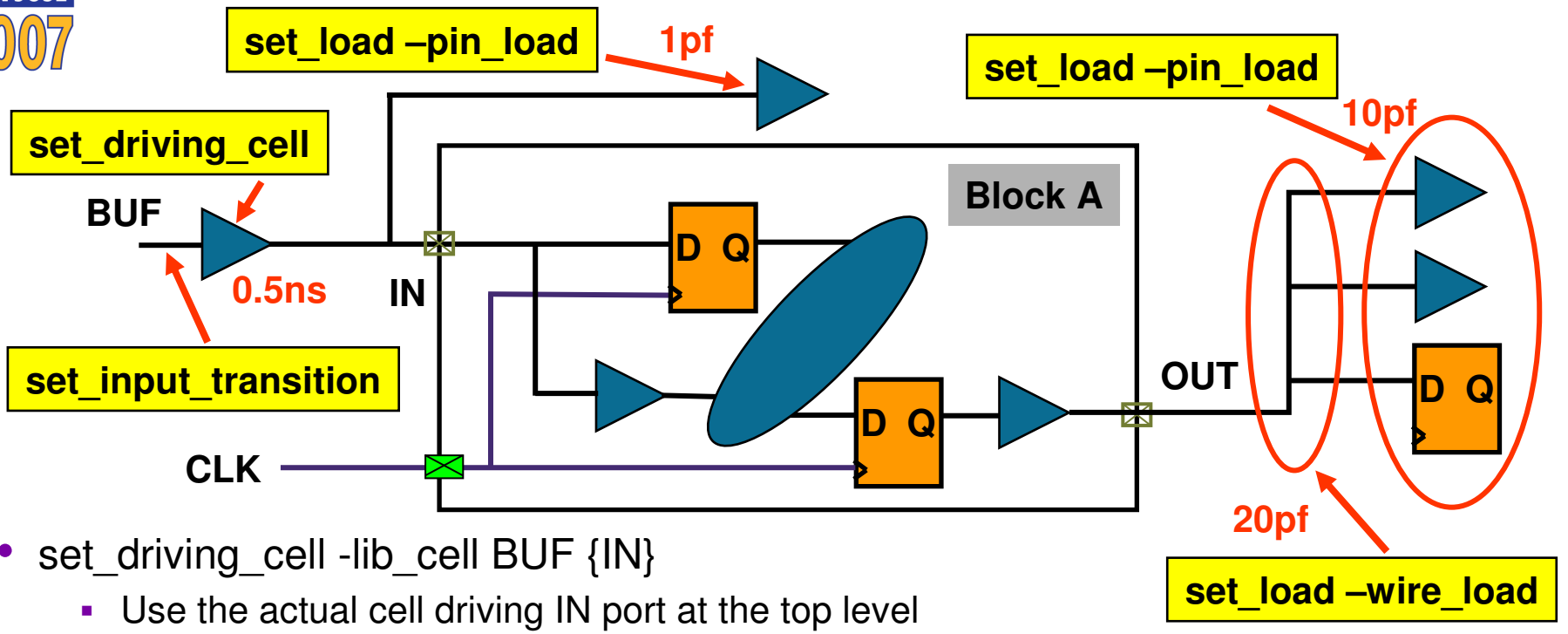
Critical Path Budgeting

- Critical path budgeting
  - JXT budgeter computes budgets (the delay constraint for each IO pin) using only the worst path
    - More efficient than all path budgeting
    - Achieve the same timing optimization QoR as all path budgeting
    - Example: for input A, for clock clk, 4 delay constraints are generated:
      - set_input_delay d1 –max –rise –clock clk –add_delay {A}
      - set_input_delay d2 –min –rise –clock clk –add_delay {A}
      - set_input_delay d3 –max –fall –clock clk –add_delay {A}
      - set_input_delay d4 –min –fall –clock clk –add_delay {A}
    - Delay d1 will be based on the worst path of max/rise condition, etc
- All path budgeting
  - Considers all the paths through the IO pins to create the constraints, which takes more memory with longer run-time

Create Synopsys Design Constraints (SDC)

- Budgeting outputs all SDC constraints required to fully describe the timing environment for blocks
  - Delay setting for IO (budget allocation)
    - **set_input_delay, set_output_delay**
  - Load/drive setting for IO
    - **set_load, set_drive, set_driving_cell**
  - Operating condition setting
    - **set_operating_conditions**
  - Timing exception setting
    - **set_false_path, set_multicycle_path, set_disable_timing**
  - etc…

Set Load/Drive for IO Pins (Ports)



- set_driving_cell -lib_cell BUF {IN}
  - Use the actual cell driving IN port at the top level
- set_load –pin_load 1 IN
  - Total pin load and driven by the same IN's driver outside of block A
- set_load –pin_load 10 –wire_load 20 OUT
  - Total pin load and wire load driven by OUT port outside of block A
- set_input_transition 0.5 IN
  - Measure the slew at the input of IN's driver

Achieve Accurate Budgeting

- More accurate timing, more accurate IO budgets
  - Need timing optimization and clock planning
- More accurate pin location, more accurate wire capacitance estimation through IO pins
  - Need pin assignment
- Better IO budgets & wire capacitance, better SDC
  - Ensure better block implementation
  - **Easier to close top-level timing after implementing blocks**

Quality of budgeting depends on quality of inputs

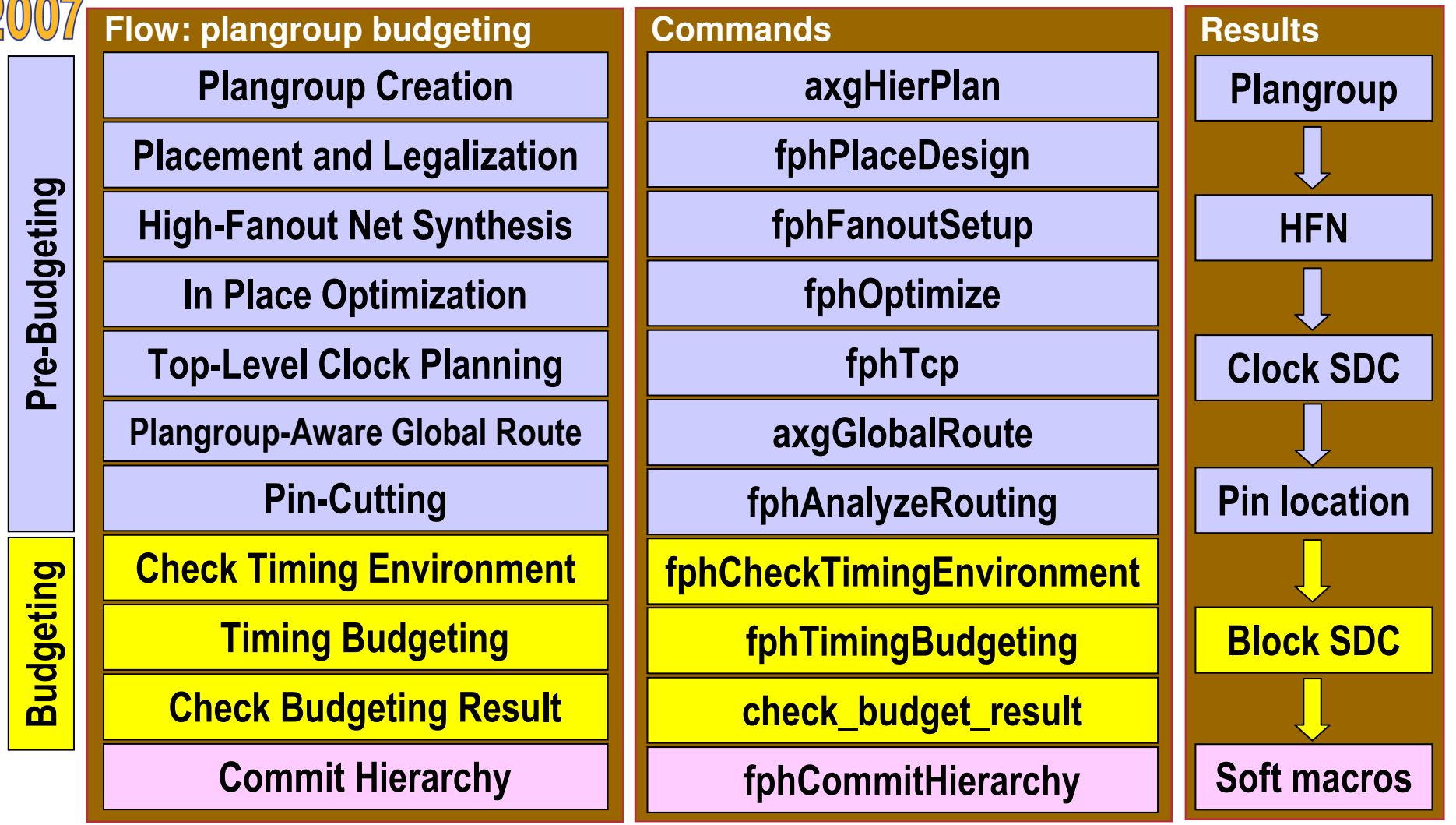
Steps To Achieve Accurate Budgeting

- Make timing more correlated to optimized design
  - Placement & legalization
  - High-fanout net synthesis
  - Full-chip in place optimization
  - Top-level clock tree planning
- Assign pin locations to well estimate wire capacitance through IO pins for each block
  - Plangroup-aware global routing
  - Pin-cutting (pin assignment based on global route)

Pre-budgeting flow: complete all the above steps

Wei-Chun Chou

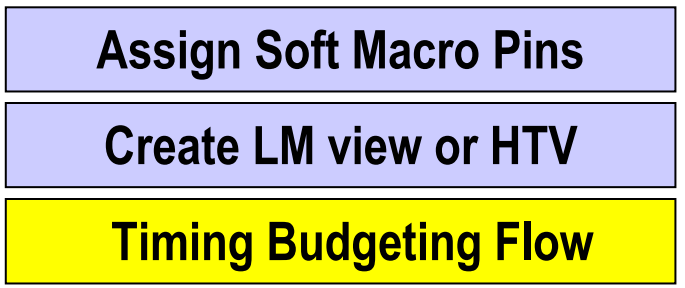
JupiterXT Pre-Budgeting & Budgeting Flow

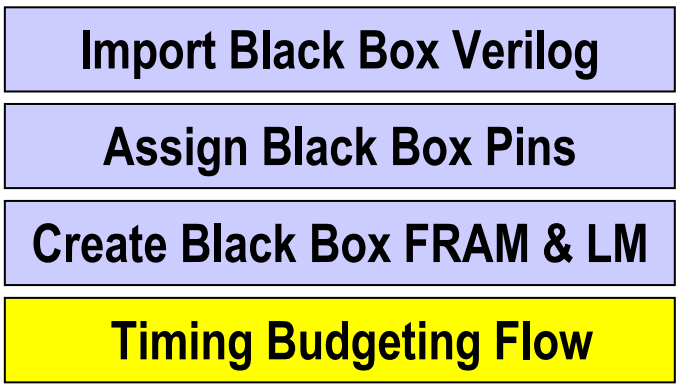


Budgeting for Prototyping

- Budgeting can be run at any stage after plangroups are created and top-level SDC is loaded
- Budgeting for prototyping
  - Budgeting is run before completion of pre-budgeting flow
  - Quickly get the idea of budgeting result, not for accuracy

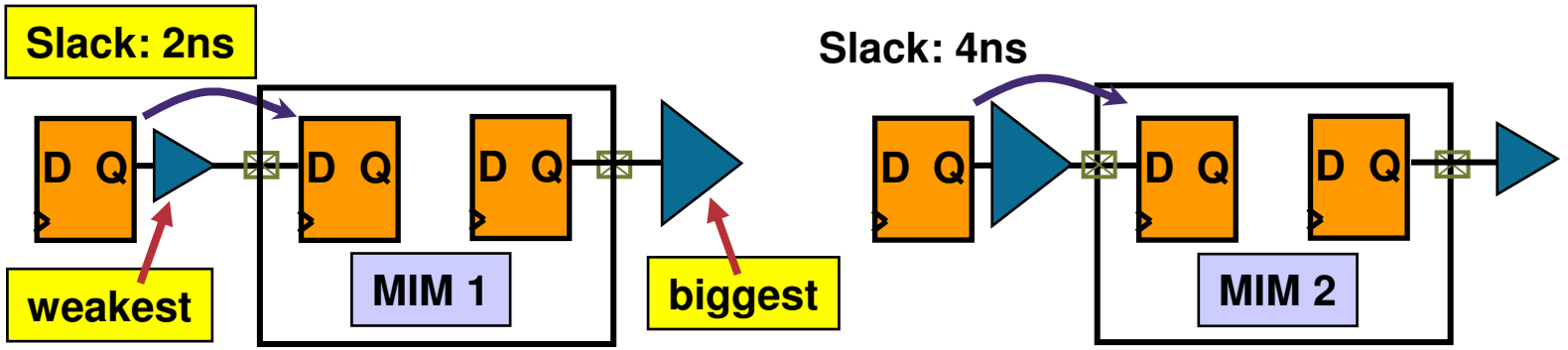
Budgeting for Soft Macros & Black Box





- Soft macro & black box budgeting
  - Special case other than recommended plangroup budgeting
  - Prerequisite
    - Create pin location on lower level CEL view
    - Create timing model (LM or HTV) for the delay of black box or soft macro

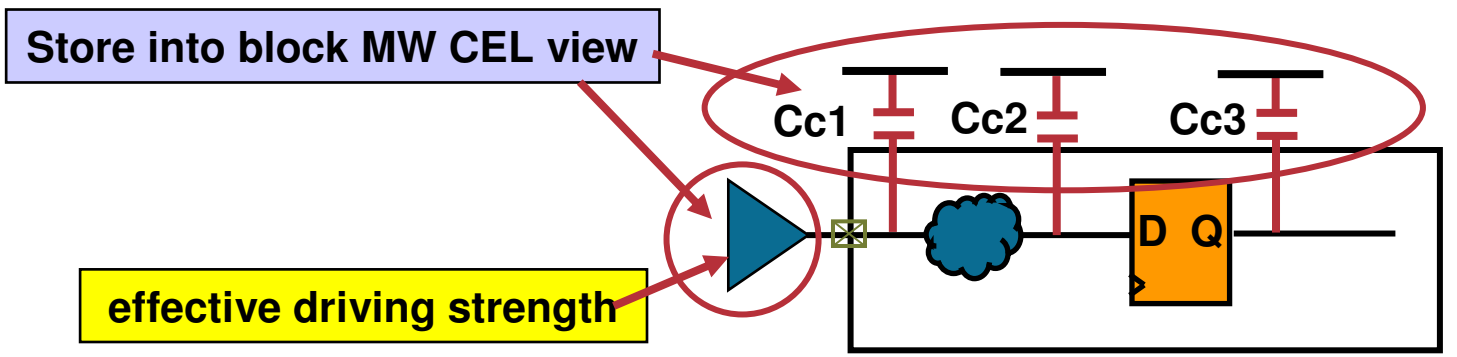
Budgeting for Multiple Instantiated Module



- Budgeting based on the worst case between MIMs
  - Specify set_input(output)_delay using the path with worst slack
    - If slacks are the same, it chooses the path with worst budget
  - Specify set_driving_cell using the weakest driver
  - Specify set_load using the biggest load
  - Generate one SDC for one MIM master
- Recommended prerequisite
  - Commit MIMs to become the same soft macro
  - Run MIM pin assignment

Budgeting Based On Crosstalk Effect

- Do budgeting using noise-induced delay
  - Click on “Enable Crosstalk Effects” in atTimingSetup
    - Timer will estimate coupling effect based on congestion map
- Store top-level xtalk effect for block implementation
  - fphSetIntParam "budgeting" “enable_hier_si" 1
    - Budgeter will store effective aggressor driving strength for input pins and coupling cap across block boundary into block CEL view



Outline

- Budgeting overview
- **Pre-budgeting flow**
  - **Placement legalization**
  - **High-fanout net synthesis**
  - **In place optimization**
  - **Top-level clock tree planning**
  - **Plangroup-aware global route**
  - **Pin-cutting**
- Budgeting flow

Placement Legalization

- Create placement and legalized all cells
- Why needed for budgeting:
  - Non-legalized placements give less accurate wire lengths
    - Wrong wire lengths generate incorrect RC values
    - Timing will be approximate, budgets will also be approximate
- Commands:
  - fphPlaceDesign, fphShapeDesign

High-Fanout Net Synthesis

- Create buffer trees for high-fanout nets
- Why needed for budgeting:
  - Reasonable wire and pin loads are created for pin on high-fanout nets in the budgeted SDC file
- Command:
  - fphFanoutSetup
- Alternative: no actual synthesis, use parameters to estimate
  - fphSetIntParam "budgeting" "fanout_load_control" 1
    - Turn on estimation, using max limit set by the following:
  - fphSetIntParam "budgeting" "max_fanout_number" <number>
  - fphSetRealParam "budgeting" "max_wire_load" <number>
  - fphSetRealParam "budgeting" "max_pin_load" <number>

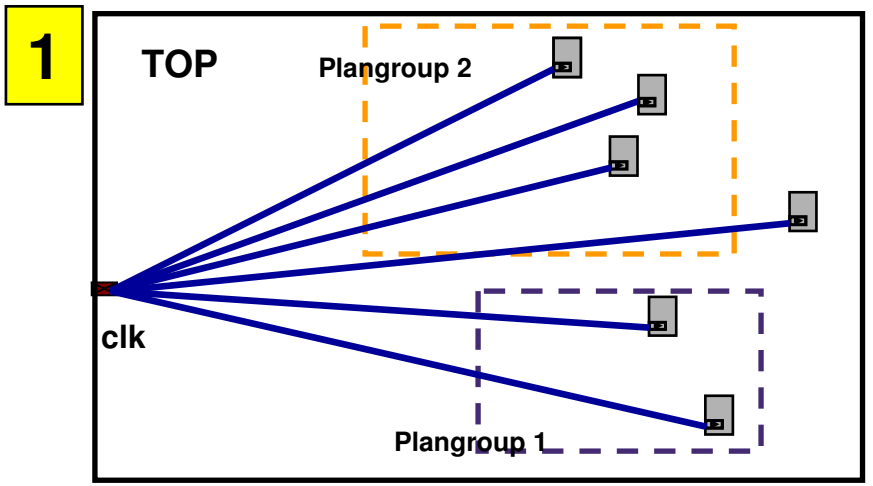
In Place Optimization (IPO)

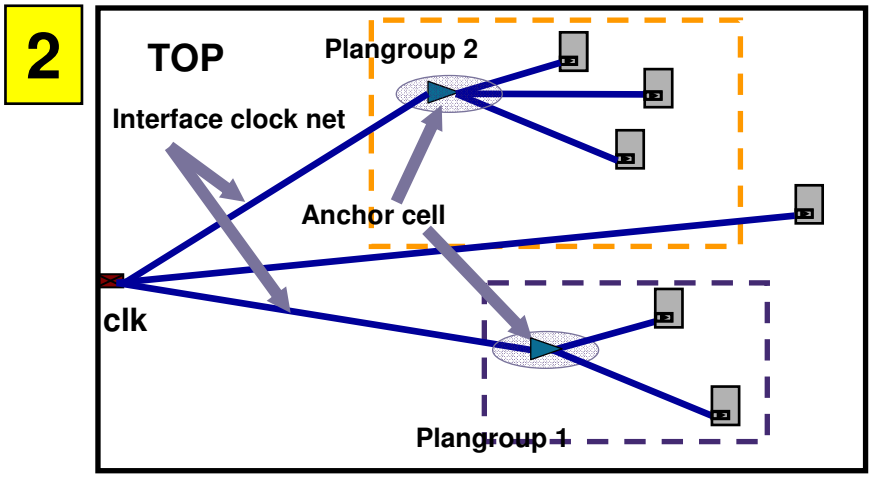
- Perform buffering and sizing to improve timing
- Why needed for budgeting:
  - Timing violations should not be more than 20% cycle time
    - If not, budgeter will produce hard-to-meet constraints
  - Use IPO to get reasonable timing delays
    - The budgeter will produce more reasonable constraints
- Command:
  - fphOptimize
- Alternative: advanced mode budgeting
  - Virtual IPO estimates IPO effect on design without changing netlist

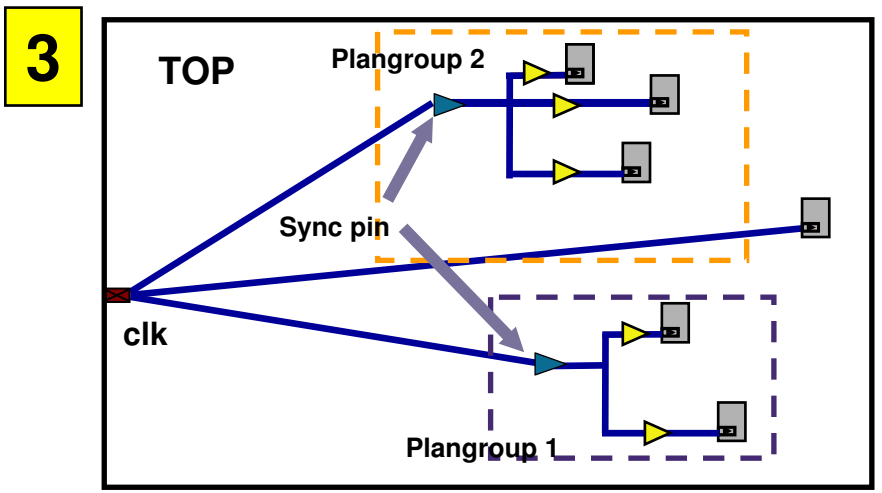
Top-Level Clock Tree Planning (TCP)

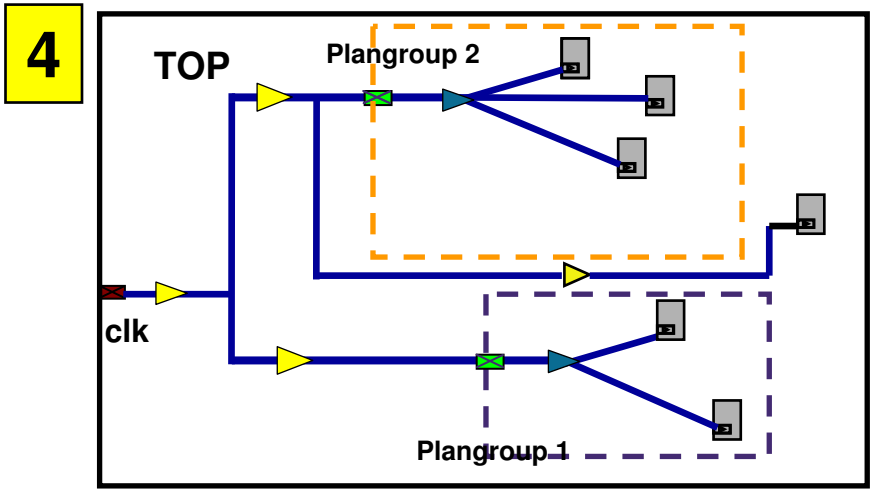
- Perform top-level clock tree planning:
  - Estimate plangroup-level clock tree
    - Insert anchor cell as tree root, run CTS
  - Implement top-level clock tree
    - Define anchor cell input as sync pin, run top-level CTS
    - Determine clock pin location
  - Create clock constraints for plangroups
- Why needed for budgeting
  - Account for real clock latency for more accurate timing
  - Create clock constraints for block-level implementation
- Commands:
  - fphClockOptions, fphAddAnchors, fphTcp

Clock Tree Planning Illustration









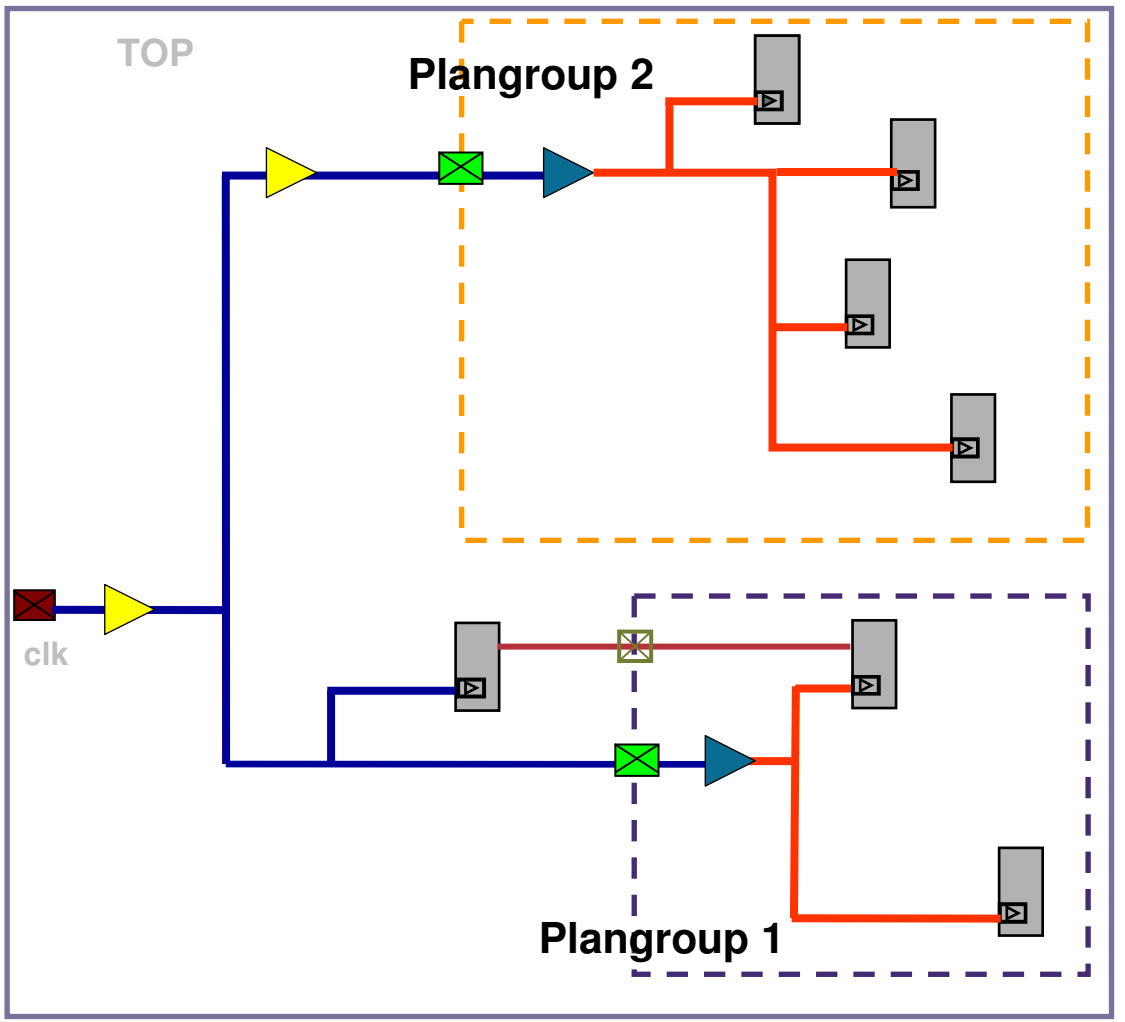
TCP Create Clock SDC

- fphTcp enables budgeter to generate clock source/network latency for each clock pin of each plangroup
  - **Clock network latency:**
    - Block-level CTS target for clock Insertion delay, for min/max, rise/fall
      - E.g. set_clock_latency 2 –max –rise [get_clock CLK]
      - Output to ./tcp_output/<plangroup>.tcp.sdc
  - **Clock source latency:**
    - Top-level clock insertion delay, for min/max, rise/fall, early/late
      - E.g. set_clock_latency 2 –source –max –rise –early [get_clock CLK]
      - Output to ./tcp_output/<plangroup>.tcp.source.sdc
- All clock latencies are stored in design milkyway such that budgeter can read and output the same latency to block SDC when running budgeting

Clock Network Latency & Source Latency

Source latency

Network latency



Who Has Clock Network/Source Latency?

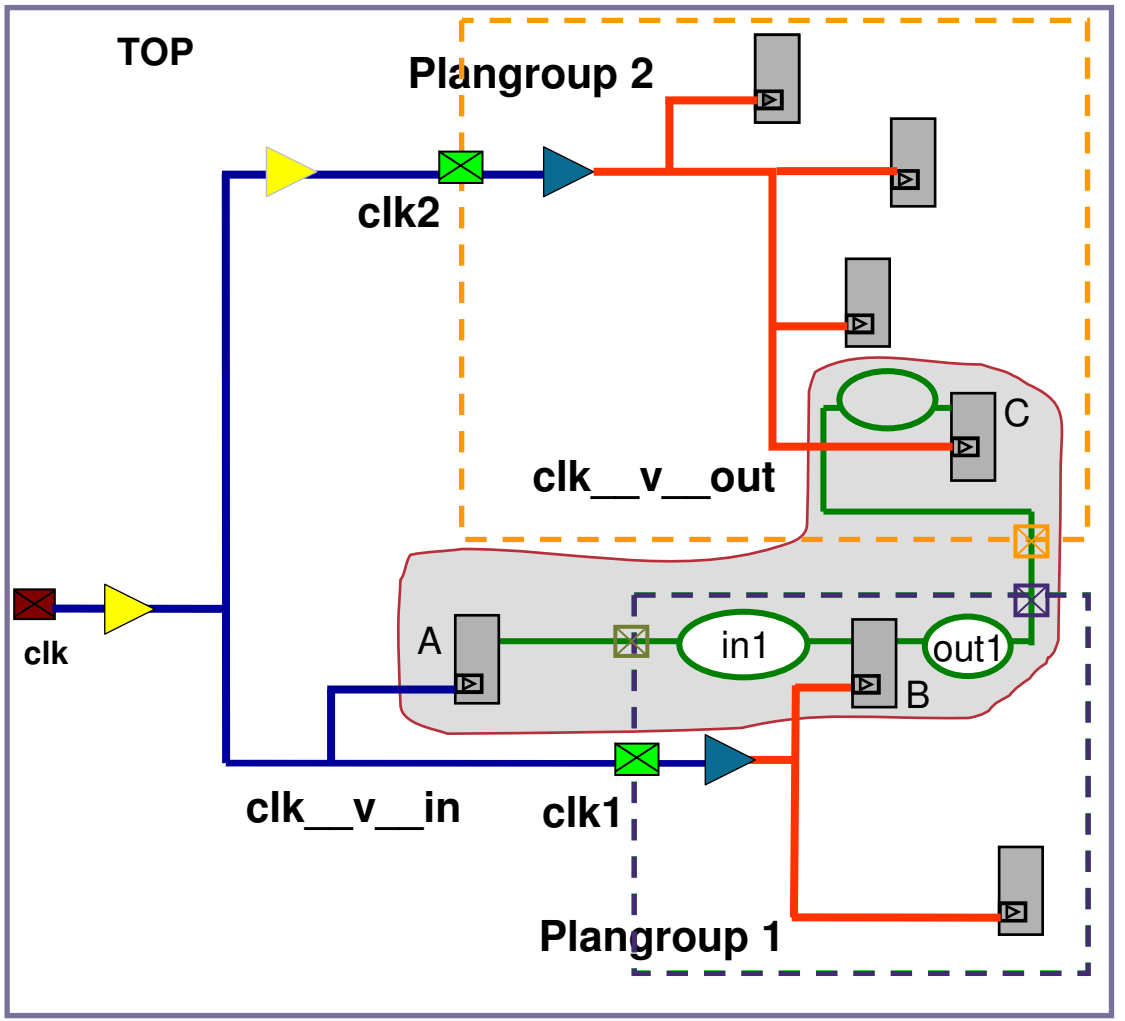
- Clocks launching or capturing flops in plangroups
  - Have both source and network latencies
- Clocks launching or capturing flops at top level
  - Have source latencies only
    - For example: a virtual clock to a plangroup

TCP Create Virtual Clock for IO Paths

- For each plangroup, virtual clock created for describing clock latency outside of the plangroup
  - Launching flops of input paths:
    - Naming convention: <clock_name>__v__in
  - Capturing flops of output paths:
    - Naming convention: <clock_name>__v__out
- To minimize the number of virtual clocks
  - Only the worst-case clock latency is created for all input/output pins of plangroups launched/captured by a virtual clock

Virtual Clock Example

- PG1 Input path in1
  - clk__v__in launch flop A on top-level: has only source latency in SDC of PG1
- PG1 output path out1
  - clk__v__out (clk2) capture flop C in PG2: has both source & network latency in SDC of PG1



Source latency

Network latency

Data latency

Virtual Clock for Input/Output Delay in Block SDC

Wei-Chun Chou

- Input and output delay statements should be expressed with respect to a virtual clock:

```
set_input_delay 2.722441 -clock [get_clocks {my_clock__v__in}] \
-rise -max -add_delay [get_ports {n856}]
```

- Exceptions:
  - Unsynthesized clocks
  - Ports connected to top-level ports.

Update Timing After TCP

- Before budgeting, full-chip timing analysis must be run to check if timing is reasonable for budgeting
- After TCP, for timer to see actual clock latency:
  - Propagate all clocks
    - tcl “set_propagated_clocks [all_clocks]”
    - In astTimingSetup, set “Ignore Propagated Clocks” off
  - For each block, annotate block-level CTS insertion delay
    - load “./tcp_output/JupiterXT.tcp.phasedelay.cmd”
      - where anchor cell input is defined as sync pin using ataDefineSyncPins and block-level clock latency is annotated at the sync pin
    - Note: JupiterXT.tcp.phasedelay.cmd is created by fphTcp

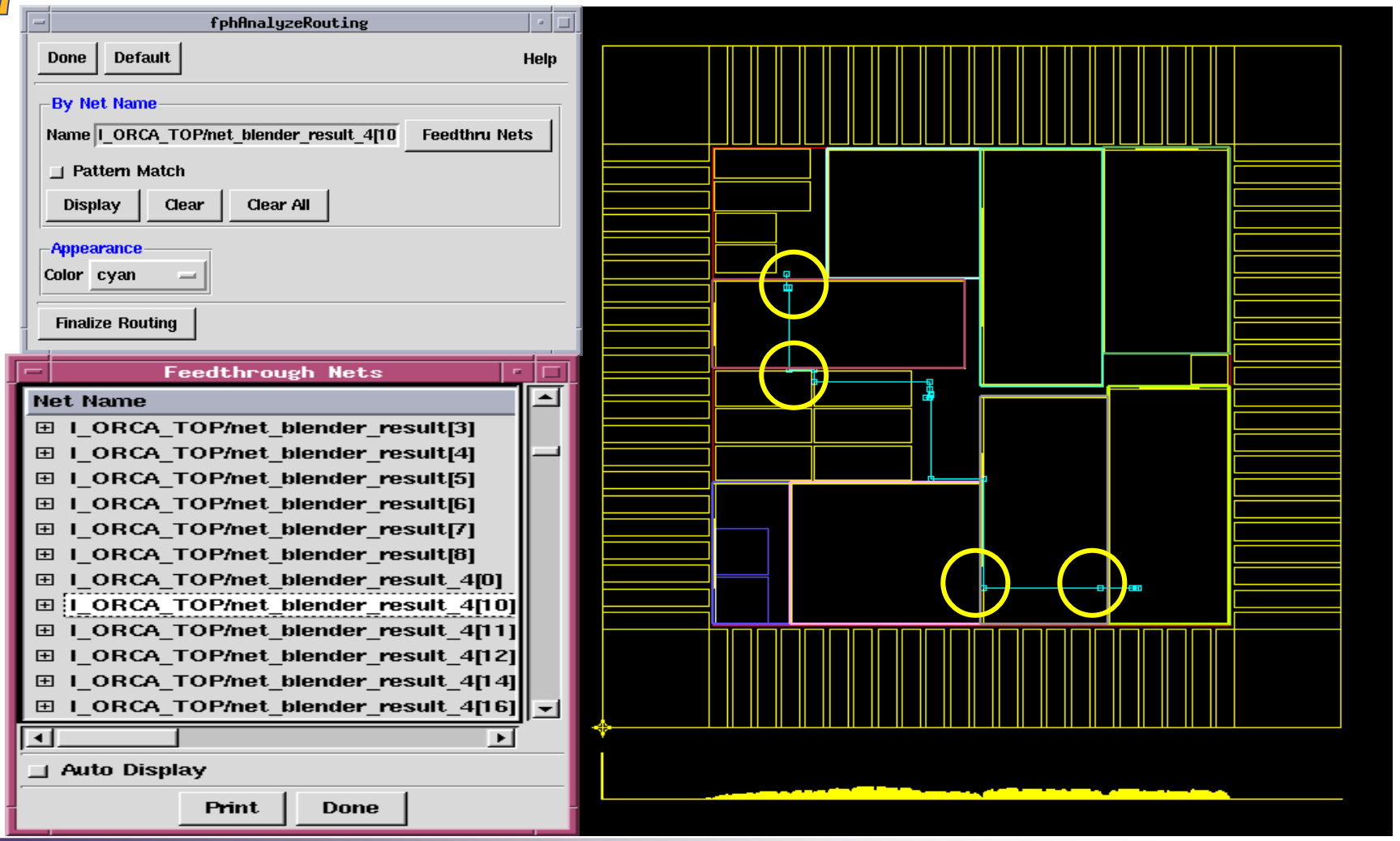
Plangroup-Aware Global Routing (PAGR)

- Perform full-chip global routing optimizing for crossing plangroup boundary (one net only route cross once)
- Why needed for budgeting:
  - Compared to virtual route, global route gives more accurate RC, hence, more accurate timing
  - Prerequisite of pin-cutting which creates pin locations for more accurate wire capacitance in SDC
- Command:
  - axgGlobalRoute or axgProtoRoute + global route option
  - axSetIntParam "route" "readPlanGroup" 1

Pin-Cutting

- Allocate pin locations on PAGR topology for each plangroup
- Why needed for budgeting:
  - Before pin-cutting, pins are at center of plangroup
  - After pin-cutting, pins got assigned to actual locations, leading more accurate timing, and the budgeter can put more accurate wire capacitance into SDC
- Command:
  - fphAnalyzeRouting

Pin-Cutting Example



Pre-Budgeting Flow Summary

- Make timing more accurate by:
  - Placement & legalization
  - High-fanout net synthesis
  - Full-chip in place optimization
  - Clock tree planning
- Assign pin to actual locations by:
  - Plangroup-aware global routing
  - Pin-cutting
- Ready for timing budgeting flow
  - More accurate budgeting is expected

Outline

- Budgeting overview
- Pre-budgeting flow
- **Budgeting flow**
  - **Check timing environment**
  - **Perform timing budgeting**
  - **Check budgeting results**

Timing Budgeting Flow

- Check timing environment
  - astTimingDataCheck
    - Check unconstrained paths
  - astTimingReport
    - Check timing is reasonable for budgeting
  - fphCheckTimingEnvironment
    - Check feasibility of the design and its timing constraints
- Perform timing budgeting
  - fphTimingBudgeting
    - Allocate budgets between blocks and create SDC for blocks
- Check budgeting results
  - check_budget_result
    - Compare actual delay with budgeted delay

astTimingDataCheck

- Check unconstrained endpoints
  - All timing endpoints in the design should be constrained
  - If not, may be due to
    - SDC error: a clock definition is missing
    - or
    - Design error: some circuits are disconnected

astReportTiming

- Check the most violating paths in the design
  - Negative slack in the design should be reasonable compared to the clock period (20% or less), or timing closure may not be achievable
  - Budgeter creates SDC with better quality if timing more correlates the final optimized implementation

fphCheckTimingEnvironment

- Provides feedback on design timing
  - Helps determine feasibility of the design and its timing constraints
- Assists in cleaning up constraint issues
  - High quality constraints generated by budgeting
- Exposes other design problems
  - E.g. dangling hierarchical pins

fphCheckTimingEnvironment GUI

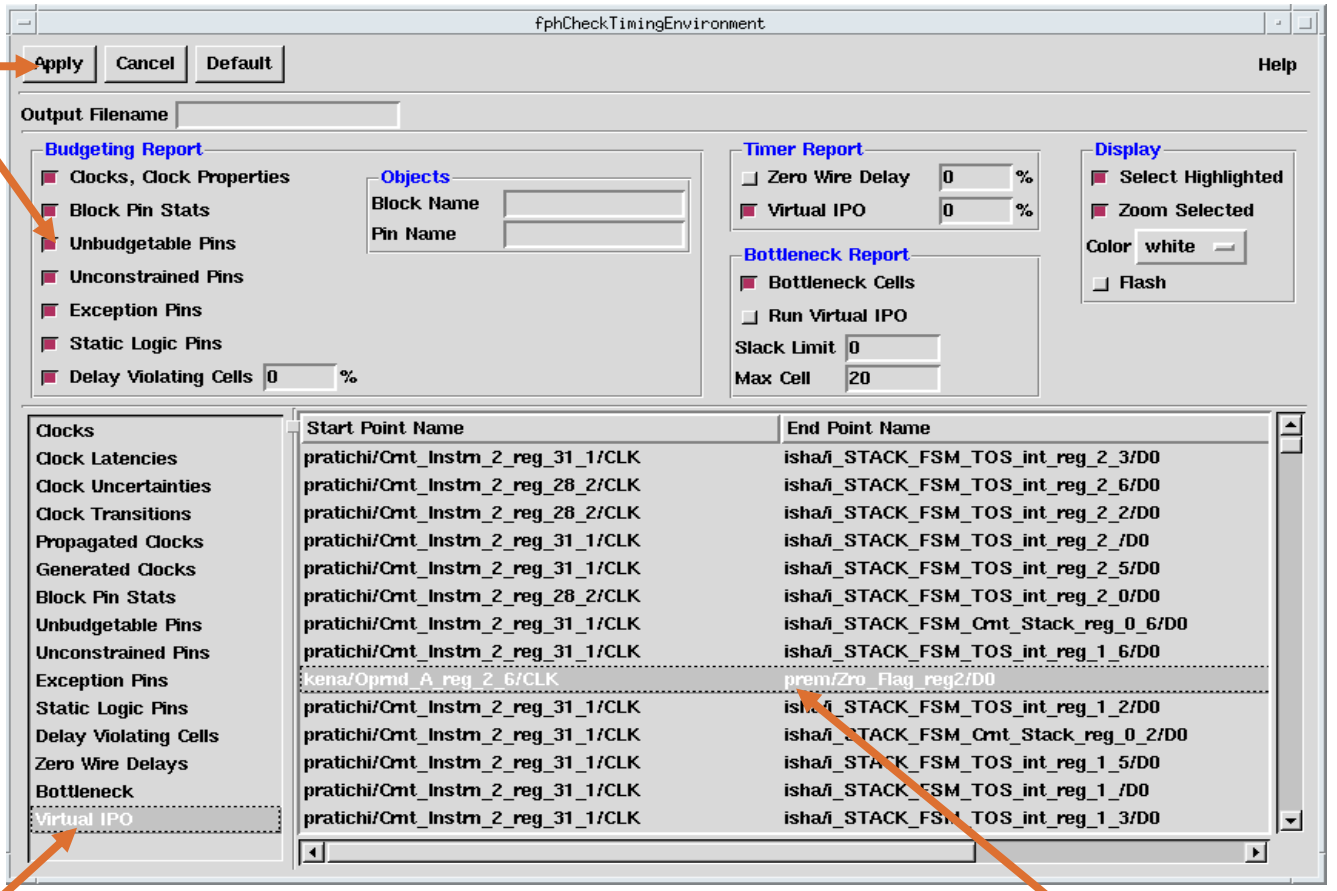
Select to apply

Unbudgetable pins:
unconnected,
unconstrained,
tie-high/tie-low,
clock, static logic,
fix delay

Exception pins:
set_false_path,
set_multicycle_path,
set_min_delay
set_max_delay

Static pins:
set_case_analysis
set_logic_one
set_logic_zero

Select report to view



Delay Violating Cells
Top level cells (on interface paths) with delays greater than specified percentage of capture clock period.
Bottleneck Cells
Print the leaf cells which contribute to multiple timing violations

Highlight in Layout GUI

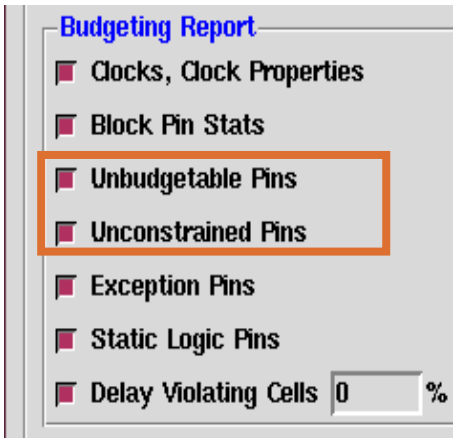
fphCheckTimingEnvironment: Budgeting Report (I)

Wei-Chun Chou

- Clocks, Clock Properties
  - Reports the clock name, driver pin, period and edges (rising and falling) of the clock.
  - Report clock latency, uncertainty, propagated and generated Clocks.
    - **Note: Check that clock definitions were read in correctly**
- Block Pin Stats
  - Prints the number of budgetable, unbudgetable and total number of pins on the block.
    - **High number of un-budgetable pins points to design problems**

fphCheckTimingEnvironment Budgeting Report (II)

Wei-Chun Chou



- Unbudgetable Pins

  Report why pins cannot be budgeted:

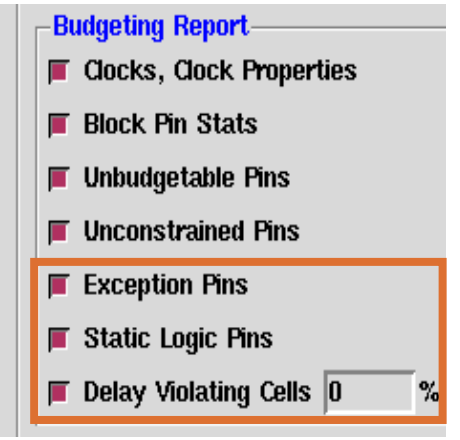
  - Pin is not connected to a net
  - Pin is not connected to a top level net, but connected to an internal net
  - Pin is not connected to an internal net, but connected to a top level net
  - Pin on paths with endpoints that are not constrained
  - Pin is a clock port
  - Pin is connected to power/ground net, or has set_case_analysis on it
  - Only fixed delay exists on one side of the pin

- Unconstrained Pins

  - Report pins with no constrained path going through

fphCheckTimingEnvironment Budgeting Report (III)

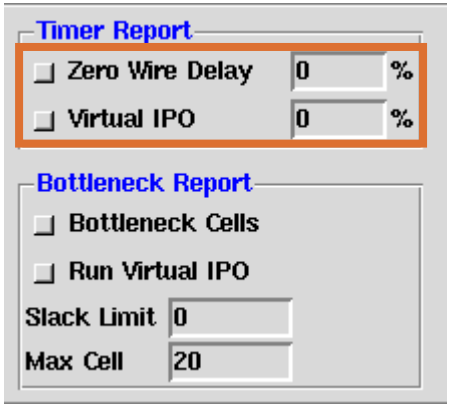
Wei-Chun Chou



- Exception Pins
  - List pins with a timing exception applied
  - Timing exceptions:
    - set_false_path, set_multicycle_path,
    - set_min_delay, set_max_delay
- Static Logic Pins
  - List pins set to static logic state by
    - set_case_analysis, set_logic_one, set_logic_zero
- Delay Violating Cells
  - List top-level cells on interface paths with delays greater than specified percentage of clock period

fphCheckTimingEnvironment Timer Report

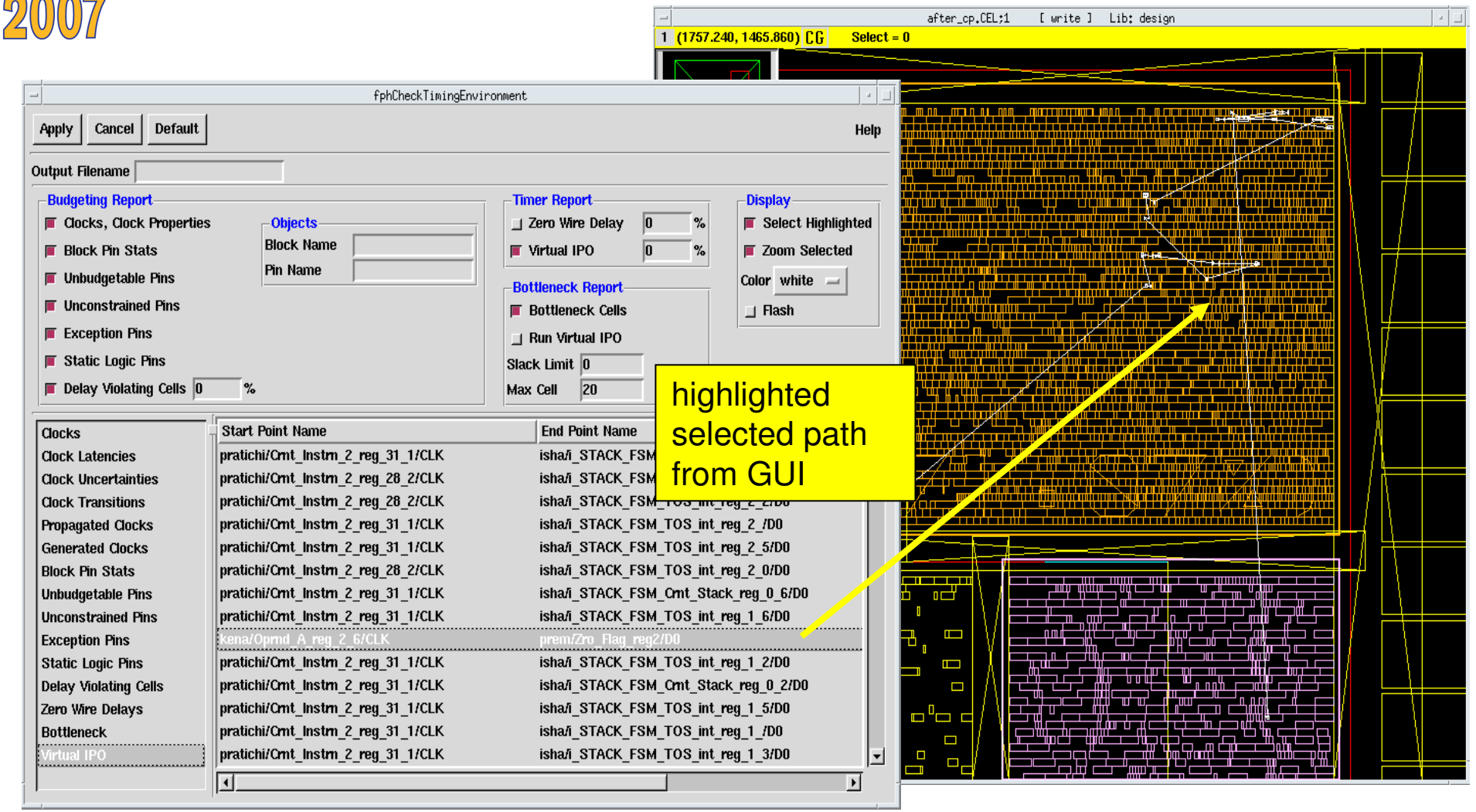
Wei-Chun Chou



- Zero Wire Delay
  - Perform zero wire delay timing analysis
  - Report the paths with the negative slack less than the cycle time percentage specified
  - Check if timing can be met without wire delay
- Virtual IPO
  - Perform timing analysis based on virtual IPO
  - Report the paths with the negative slack less than the cycle time percentage specified
  - Check if timing closure can be achieved by real IPO
- Zero Wire Delay and Virtual IPO will be run in two separate timing analysis sessions
- Original timing will be restored after reports are done

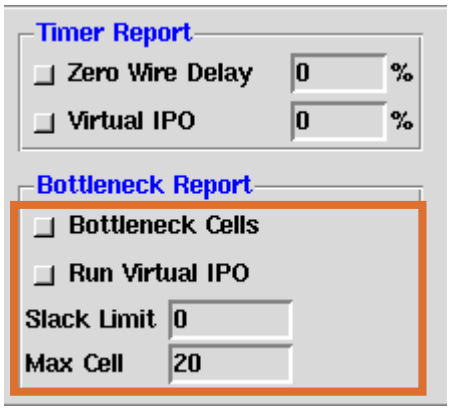
fphCheckTimingEnvironment Timing Path Highlight

Wei-Chun Chou



fphCheckTimingEnvironment Bottleneck Report

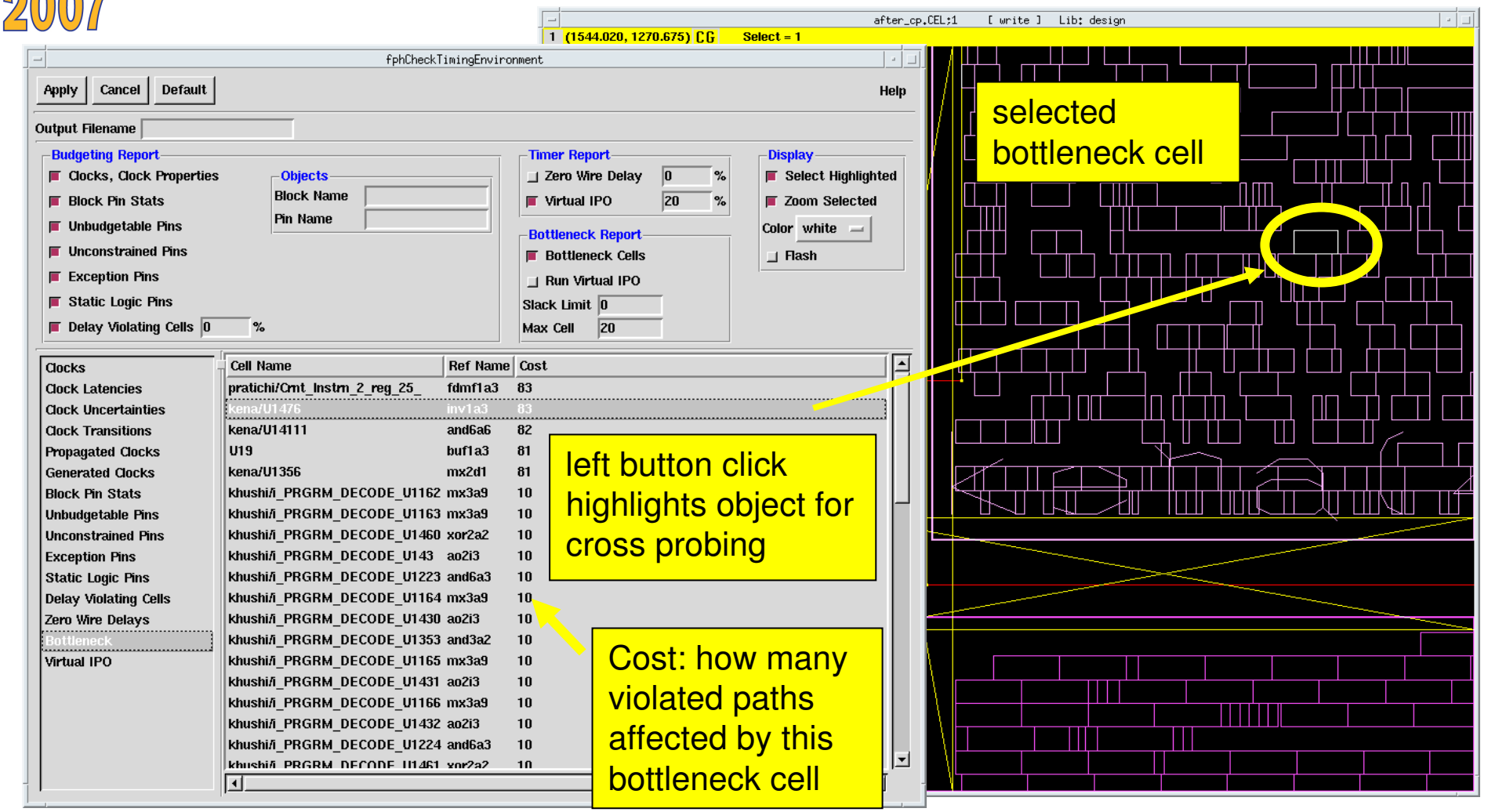
Wei-Chun Chou



- Bottleneck Cells
  - The leaf cells which contribute to multiple timing violations for entire design
    - **NOTE: Check these cells for potential problems**
- Run Virtual IPO
  - Report bottleneck cells based on virtual IPO analysis
- Slack Limit
  - Only the paths with slack less than the slack limit will be checked for locating bottleneck cells
  - If set to run virtual IPO, IPO-based timing will be used for slack
- Max Cells
  - Maximum number of bottleneck cells to be reported

fphCheckTimingEnvironment Bottleneck Cells – Cross Probing

Wei-Chun Chou



fphCheckTimingEnvironment Recommended Flow

Wei-Chun Chou

1. Report timing with Zero Wire Delay (if real IPO not run)
   - Overview the top-level timing
   - Point out potential design problems
2. Report timing with Virtual IPO (if real IPO not run)
   - Quick feasibility analysis to show if violating paths are fixable
3. Report Bottleneck Cells
   - Report cells contributing to most violating paths
4. Report Block Pin Stats
   - Show how many pins can be budgeted
5. Report Unbudgetable Pins
   - List unbudgetable pins and show the reason why unbudgetable
6. Run other budgeting reports to analyze plangroup in detail

Check Timing Environment Summary

- Check timing for entire design
  - ataTimingDataCheck
    - Check unconstrained paths
  - astReportTiming
    - Check if timing is reasonable (negative slack < 20% cycle time)
  - fphCheckTimingEnvironment with Zero Wire Delay or Virtual IPO
    - Check potential design problem affecting timing closure
  - fphCheckTimingEnvironment with Bottleneck Cells
    - Check for cells contributing multiple timing violations
- Check timing for plangroups
  - fphCheckTimingEnvironment with Block Pin Stats or Unbudgetable Pins
    - Check for pin status for budgeting per block
- Resolve identified problems before budgeting

fphTimingBudgeting

- Three approaches to allocate budgets for blocks
  - **Traditional approach**
    - Compute budgets proportionally to existing block delay
  - **Advanced approach**
    - Compute budgets proportionally to block delay based on virtual IPO
  - **Set user-defined budgets**
- Create SDC for each block
  - Set input/output delay
  - Set driving cell for input ports
  - Set load for input/output ports
  - Set input transition for the driver of input ports
  - Set operating conditions
  - Set timing exceptions

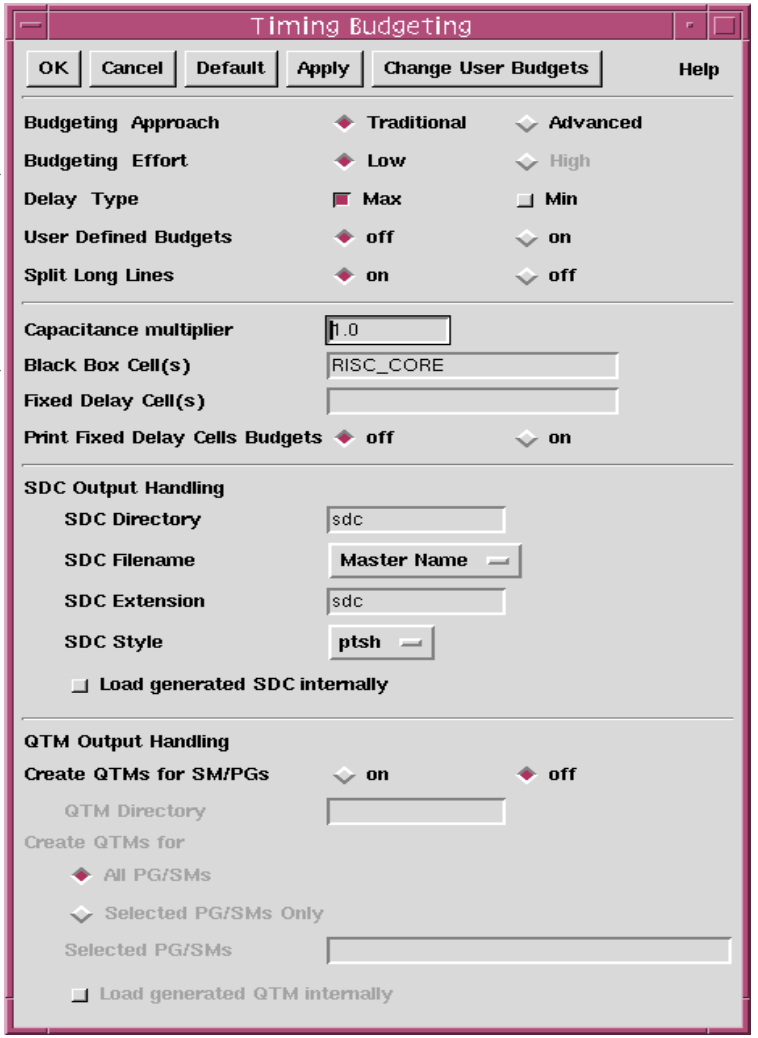
fphTimingBudgeting GUI

Approach: traditional or advanced
VIPO effort: advanced approach only
high: 2 loops, low: 1 loop
Budgeting for max/min corner
Turn on/off user-defined budgets

Use cap multiplier to scale load in SDC
Specify Black Box for budgeting
BB is Fixed Delay Cell if not specified
Specify PG/SM as Fixed Delay Cells
Actual delay allocated
Still output SDC for Fixed Delay Cells

Directory for SDC
Filename for SDC: master or Instance
Set master by default if detect MIMs
Load SDC to CEL

Create QTM for PGs or SMs
Based on budgets, perform quick top-level timing analysis before real block implementation



Advanced Budgeting Approach

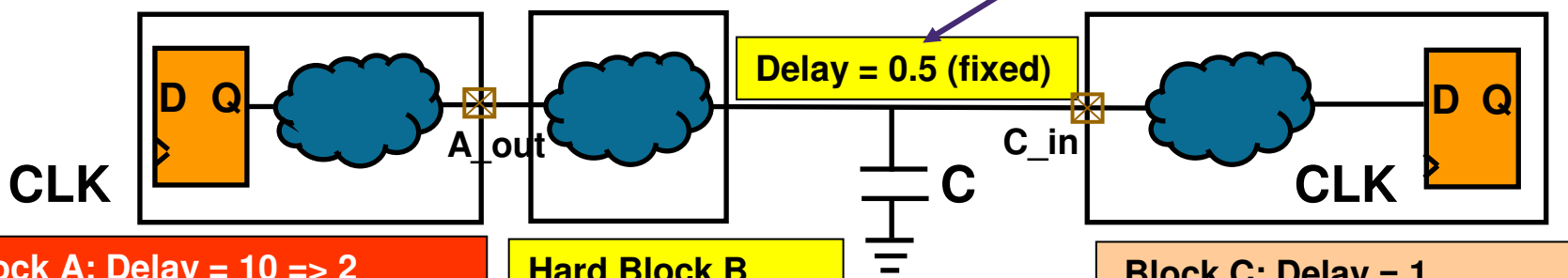
- Budgeter run virtual IPO to estimate possible timing improvement at later stage
  - Calculate buffering and sizing effects on timing virtually in memory without actually changing the netlist
  - Apply on design with bad initial timing to achieve more reasonable budgeting
- Budgeter allocates budget proportionally to the delay estimated by virtual IPO
- No need to run IPO prior to this step

Advanced Budgeting Example

Virtual IPO estimates block A can be optimized to 2ns

```
CLK cycle time = 12
Total budgetable delay = 12 - 2.5 - 0.5 = 9
Original slack = 9 - 10 - 1 = -2
Virtual IPO slack = 9 - 2 - 1 = 6
Virtual budget slack = 9 - 9 = 0
```

Top-level delay is fixed delay



Block A: Delay = 10 => 2
Virtual Budget = 9 (2/(2+1)) = 6

Hard Block B
Delay = 2.5 (fixed)

Block C: Delay = 1
Virtual Budget = 9 (1/(2+1)) = 3

Block A SDC file :

set_output_delay 5.5 –clock CLK [get_ports A_out]

output_delay = 2.5 + 0.5 + 3 = 5.5

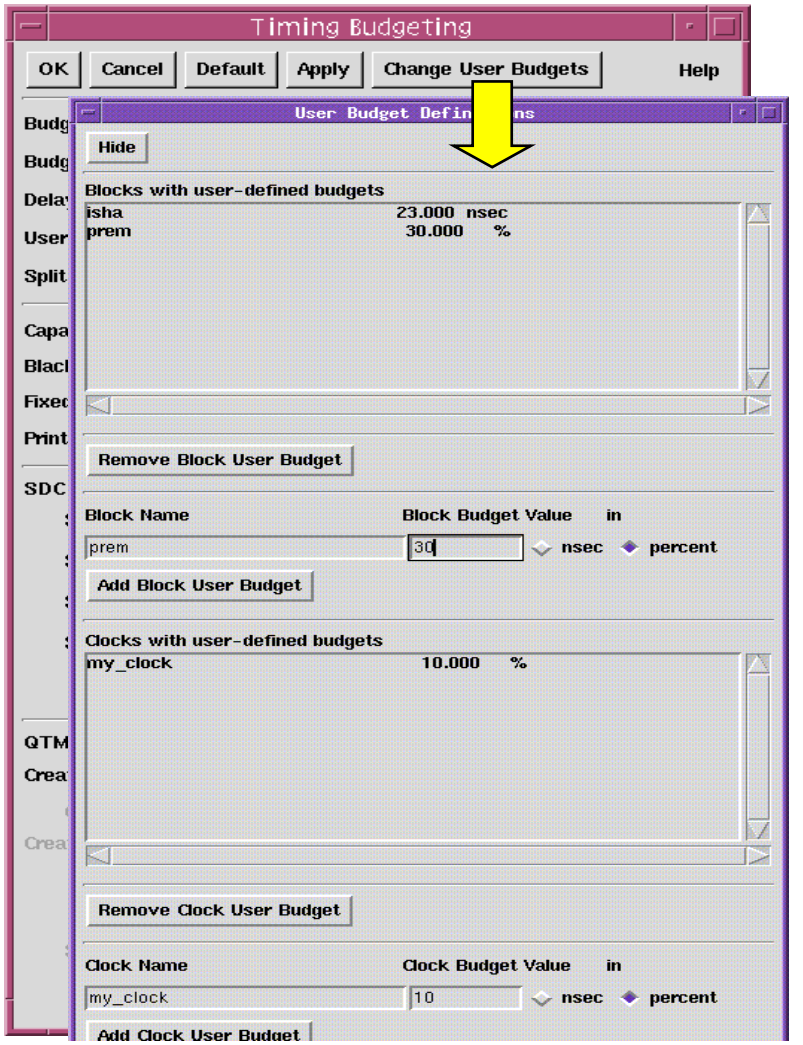
Block C SDC file:

set_input_delay 9 –clock CLK [get_ports C_in]

input_delay = 6 + 2.5 + 0.5 = 9

User-Defined Budgets

- Block-based user-defined budgets
  - **For each block**, specify the budget as a fixed delay or % of total budgetable delay
  - Specify the block name
  - Click on “Add Block User Budget”
- Clock-based user-defined budgets
  - **For each clock**, specify the budget as a fixed delay or % of total budgetable delay
  - Specify the clock name
  - Click on “Add Clock User Budget”
- If user-defined budgets plus top-level delay exceed clock cycle time
  - User-define budget is not used
  - Normal proportional budgeting is used

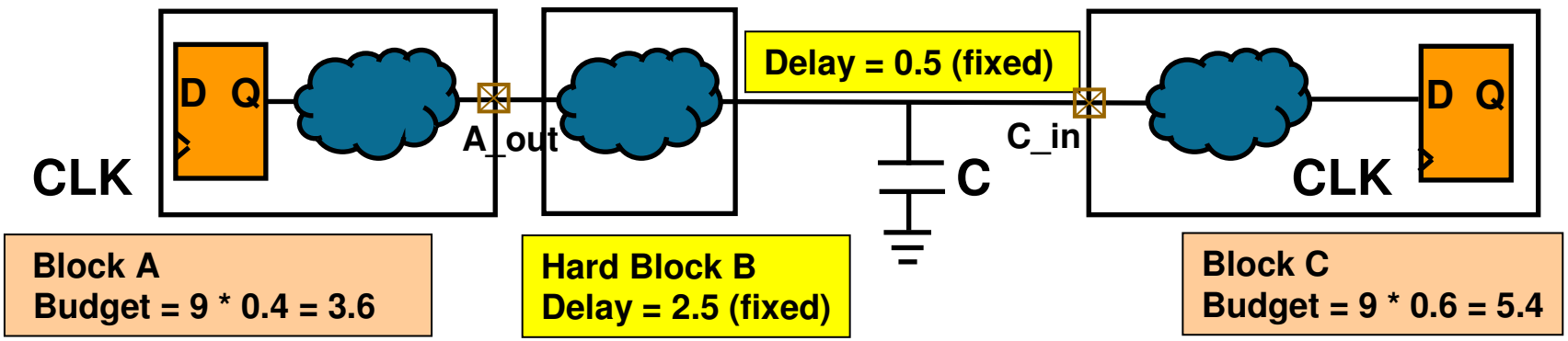


User-Defined Budget Example: Block-Based Percentage

Wei-Chun Chou

Budget of Block A = 40% * Total Budgetable Delay
Budget of Block C = 60% * Total Budgetable Delay

```
CLK cycle time = 12
Total budgetable delay = 12 - 2.5 - 0.5 = 9
```



Block A SDC file :

set_output_delay 8.4 –clock CLK [get_ports A_out]

output_delay = 2.5 + 0.5 + 5.4 = 8.4

Block C SDC file:

set_input_delay 6.6 –clock CLK [get_ports C_in]

input_delay = 3.6 + 2.5 + 0.5 = 6.6

set_context_margin

- Tighten or relax the input/output delay constraints created by the budgeter in block implementation
  - Based on user's design knowledge to each IO requirement
  - Give more flexible budget control per IO pin for early netlist
- It is a TCL command, usage:

  set_context_margin
  - [-percent] (consider specified value as a percentage of the delay)
  - [-relax] (relax instead of tightening the constraints)
  - [-min] (specify the margin for hold constraints)
  - [-max] (specify the margin for setup constraints)
  - value (determine the margin in absolute or percentage value)
  - [list] (indicate a list of cells or pins)

Warnings During Budgeting

- Warnings point out fundamental timing problems in the design
- Warnings may indicate that timing closure is not possible using block-level optimization
- Example
  - WARNING : Total fix delay (350.356) is greater than required time (330.843), I/O delay will be adjusted for port `portA' in cell `BlockA'.
  - **Source of problem: primary input pin with input delay greater than clock cycle defined in SDC**

Debug Variable – Input/Output Delay

- Find the timing path used in budgeting:
  - fphSetIntParam “budgeting” “debug_level” 1
- In SDC, before each set_input(output)_delay, it prints start point and end point of the corresponding worst path used for allocating budgets, as a comment
  - Use timing report to check full path

check_budget_result

- Post-budgeting analysis:
  - Report and compare the actual and budgeted path delays on the interface paths for each block
    - Flop-to-flop paths in blocks will not be reported
- It is a TCL command, usage:

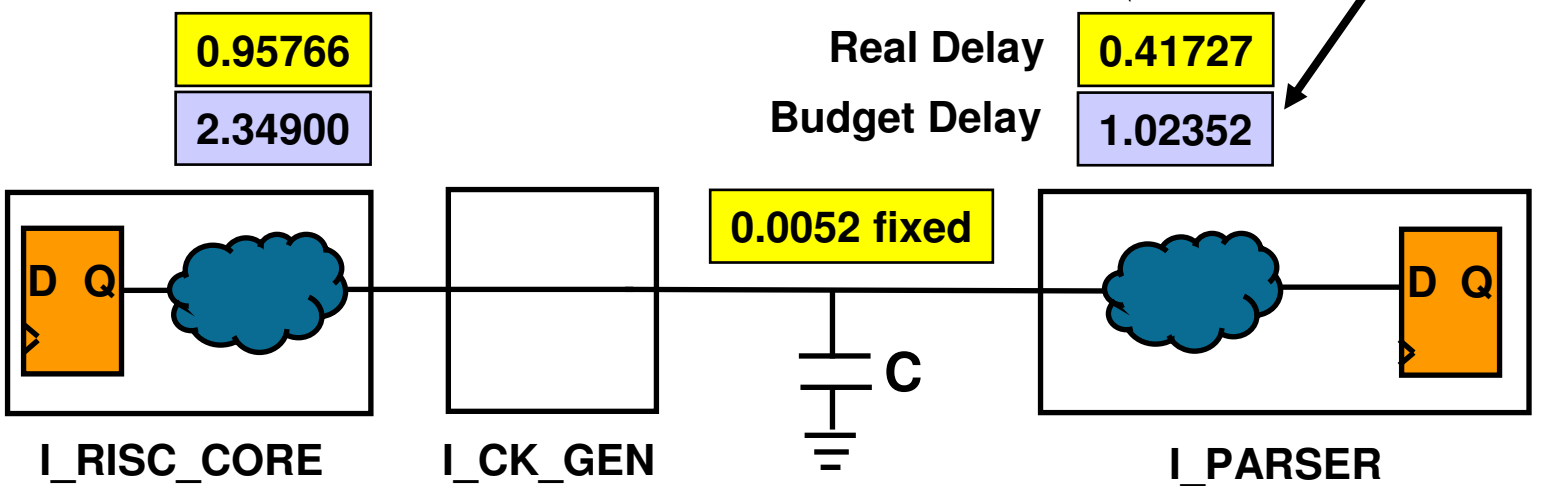
  check_budget_result
    - [-block_name string] (name of one block instance)
    - [-pin_name string] (name of one pin on block instance)
    - file name (name of output report file)
- Must be run during the same session in which budgeting is run

Post-Budget Analysis Report

```
Timing path starting point: I_TOP/I_RISC_CORE/\I_INSTRN_LAT_Crnt_Instrn_1_reg[2]/CP

                                                          Real Delay  Type    Budget Delay
                                                          ==========  ====    ============
I_TOP/I_RISC_CORE/\I_INSTRN_LAT_Crnt_Instrn_1_reg[2]/CP     0.00000             0.00000
I_CK_GEN/I_TOP/net_risc_Xecutng_Instrn[2]__pft              0.95766             2.34900
I_CK_GEN/I_TOP/net_risc_Xecutng_Instrn[2]__pft1 <--         0.00520   fixed     0.00520
I_TOP/I_PARSER/\i_reg_reg[4]/D                              0.41727             1.02352
Timing path ending point: I_TOP/I_PARSER/\i_reg_reg[4]/D

End Point Skew: 0.25178     Path delay : 1.38013      Required Time : 3.37772      Fixed
Delay : 0.00520
```



Summary

- Budgeting is to generate SDC for implementing blocks in hierarchical flow
  - During timing analysis and SDC creation, budgeter can early detect the feasibility of timing closure
- Budgeter needs pre-budgeting flow to achieve good estimation for budgeting
  - Quality of budgeting depends on quality of inputs
- Users run budgeting flow to check budgeting environment, invoke budgeter to create SDC, and check final budgeting results

Thank You