The Arrow of Time: Following Timing Constraints in an RTL to GDSII Flow

Dwight Hill
Synopsys

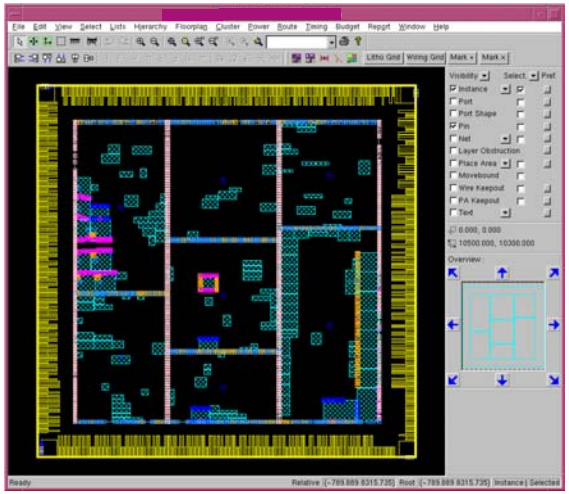
Outline

- **Intro and Background**
- What Are Timing Constraints?
- Formats for Timing Constraints
- Hierarchy Mapping
- Time Budgeting
- Eco and other technology issues
- Summary

A tutorial: not a product announcement

nVIDIA

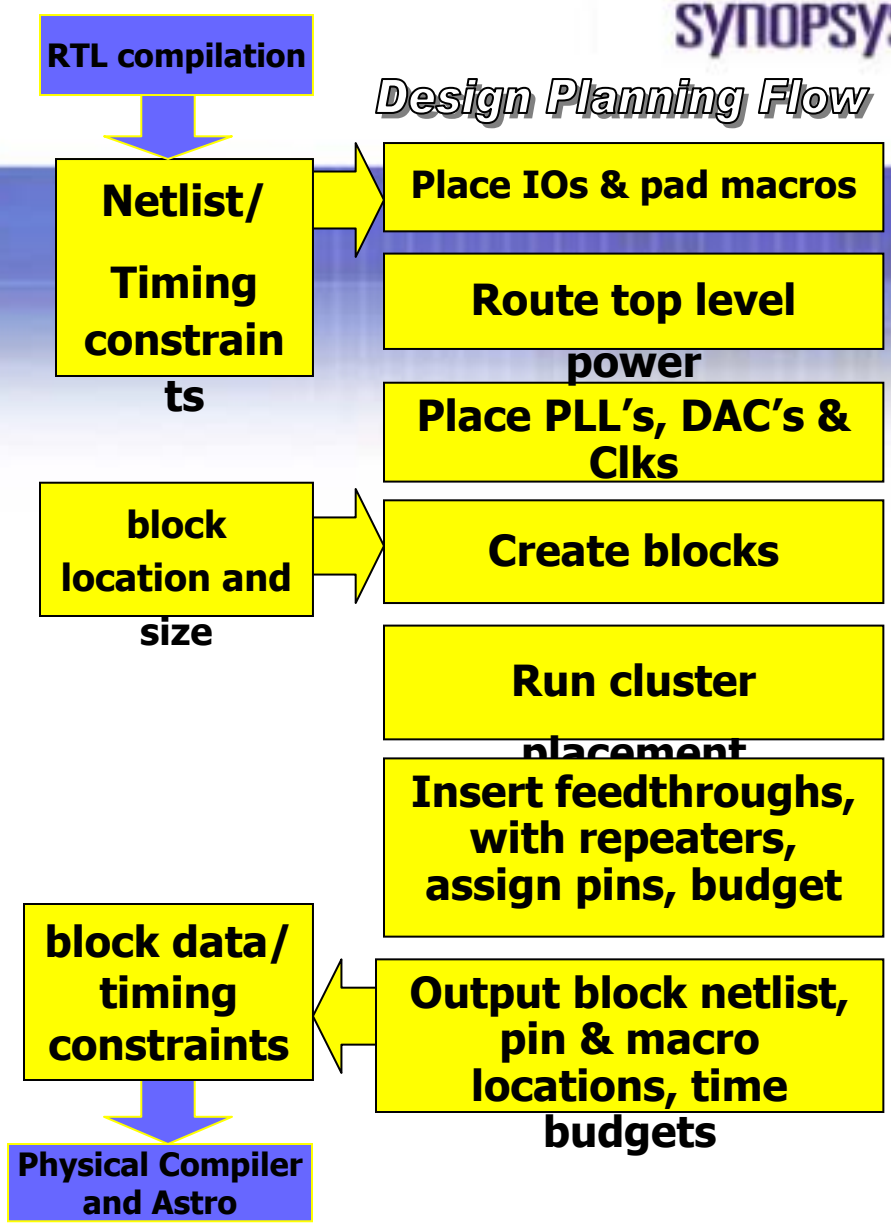
# Gates	10 M
# Instances	2.1 M
# Blocks	7
# Hard Macros	120
# IO's	500
Clock Speed	500 MHz



Floorplan after Pinning, Feeds, and Repeater Insertion

SYNOPSYS

Design Planning Flow



What are Timing Constraints?

SYNOPSYS

- Synchronous circuits have timing defined by one or more clocks
  - (normal, generated, virtual)
- But most of the complexity of constraints is in timing exceptions:
  - Paths that will not be valid during a timing run
  - May reflect diagnostic, chip initialization, or undefined logical behavior.

Formats for Constraints

SYNOPSYS

- Most constraints are in Synopsys Design Constraint (SDC) or similar format
  - SDC == subset of Tool control Language (Tcl)
  - Objects are referred to by
    - Name (with implied type)
      - Set_false_path –to nand23/in
    - “get_object” commands
      - Set_false_path –to [get_pins nand23/in]
    - Collection handles
      - Set_path_path –to $nand_pins
    - Procedure

      Set_false_path –to [find_slow_pins]

SDC examples

SYNOPSYS

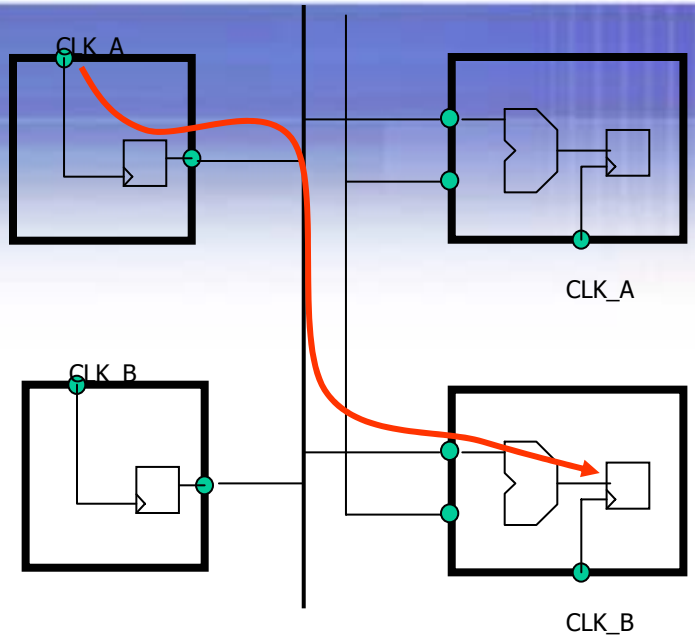
```
set_false_path -from [get_pins blocka/out*] -to [get_pins blockb/in*]
```

```
proc set_tc {fromblockname  toblockname} {
   foreach inp [get_pins -filter "direction==in &&
     type==signal" -of $from_blockname]  {
       foreach outp [get_pins -filter "direction==out" -of $to_blockname]{
            set_false_path -from $inp -to $outp;
            }
 }

# now invoke the proc
set_tc blocka blockb
```

Both representations may have the same effect

False Paths



This path could be disabled in one of several ways, e.g.

```
set_false_path -from [get_clocks clk_a] -to [get_clocks clk_b]
```

or

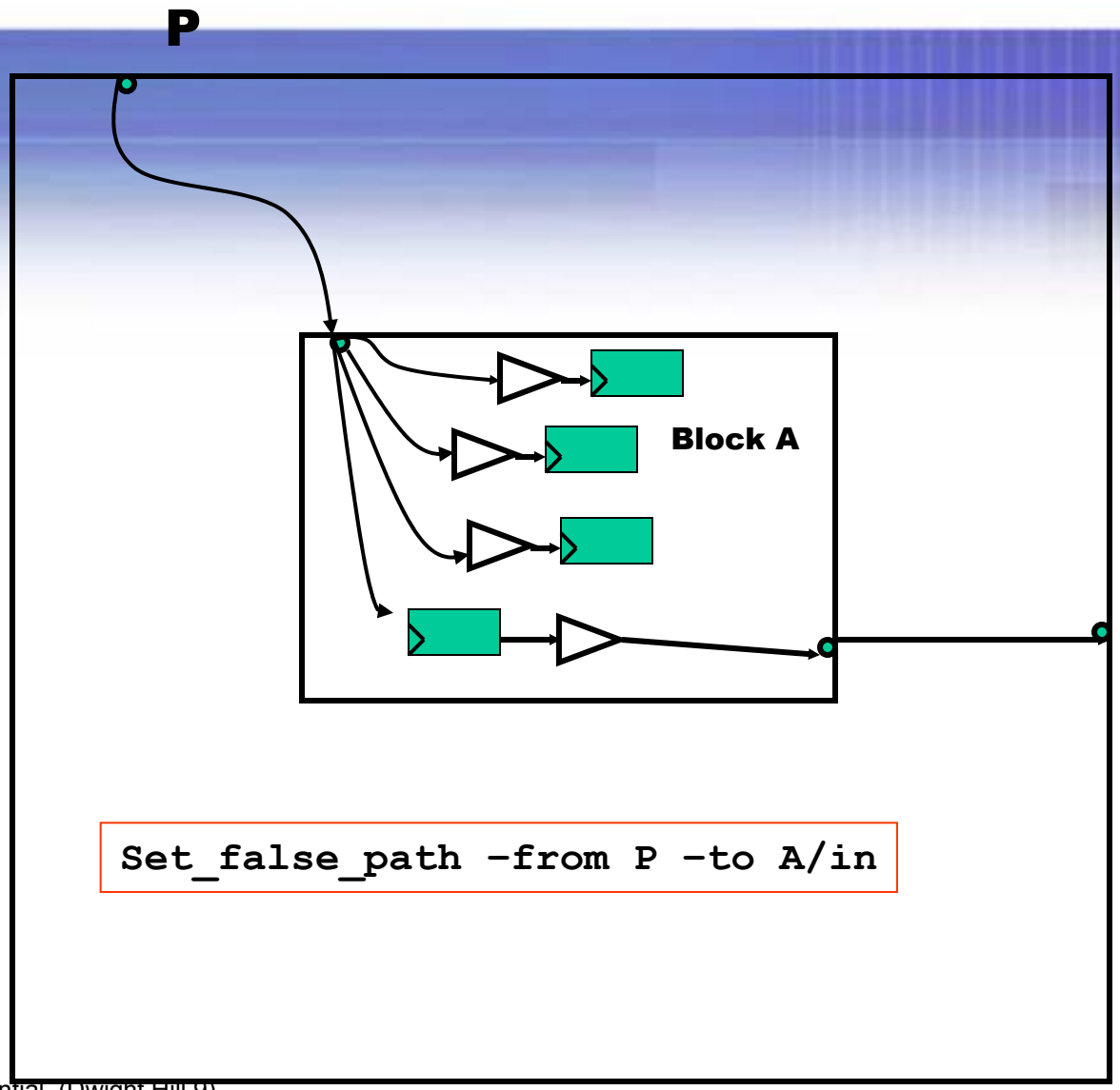
```
set_false_path -from reg_a_left.out -to reg_b_right.in
```

Hierarchy Mapping

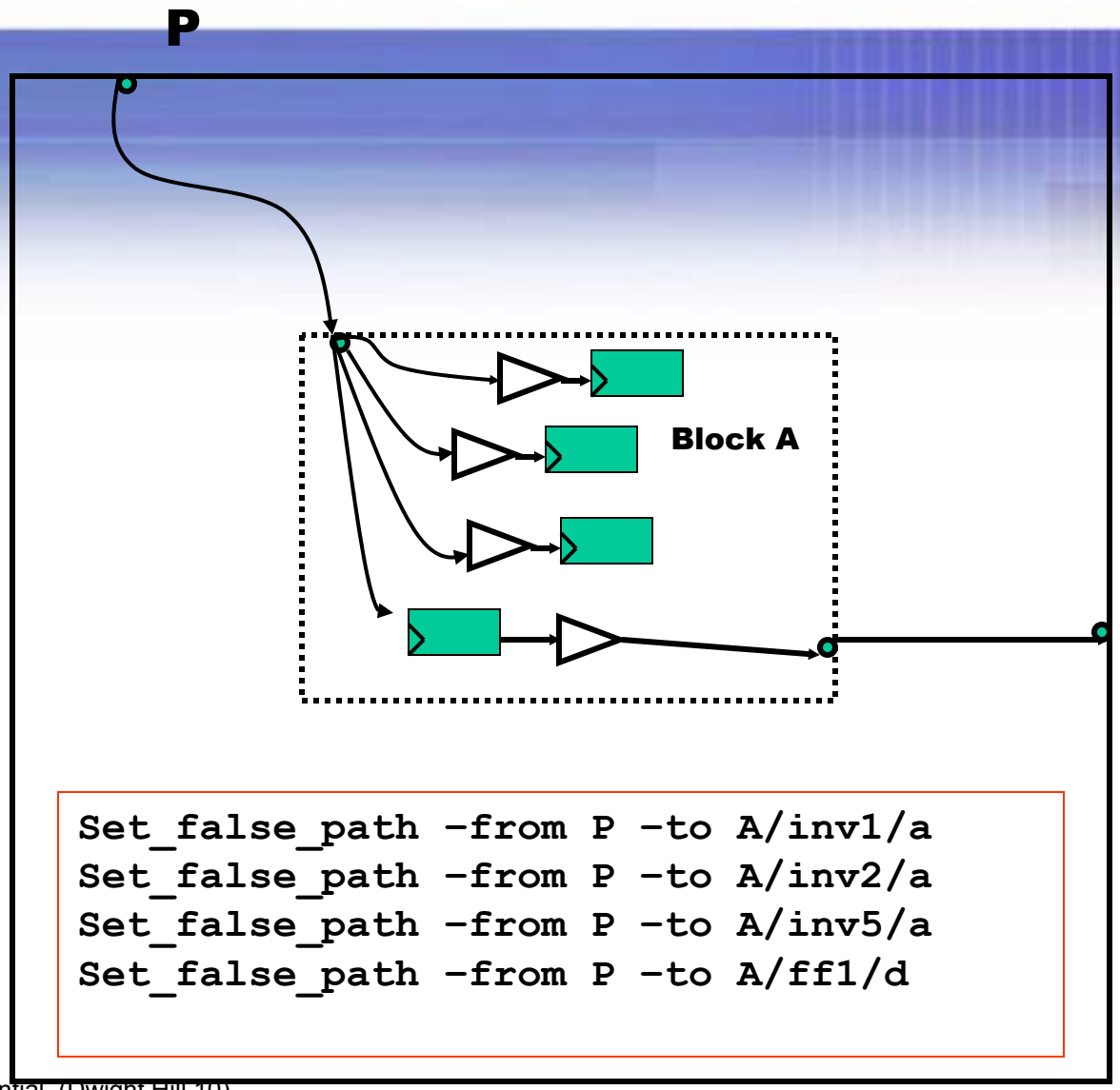
SYNOPSYS

- Hierarchy
  - Logical Designs have deep and varied hierarchy
  - Physical designs are flat, or at least shallow
  - Timing constraints written on logic hierarchy
    - May refer to pins on blocks that are flattened (and hence no longer exist)

Hierarchical Contraints



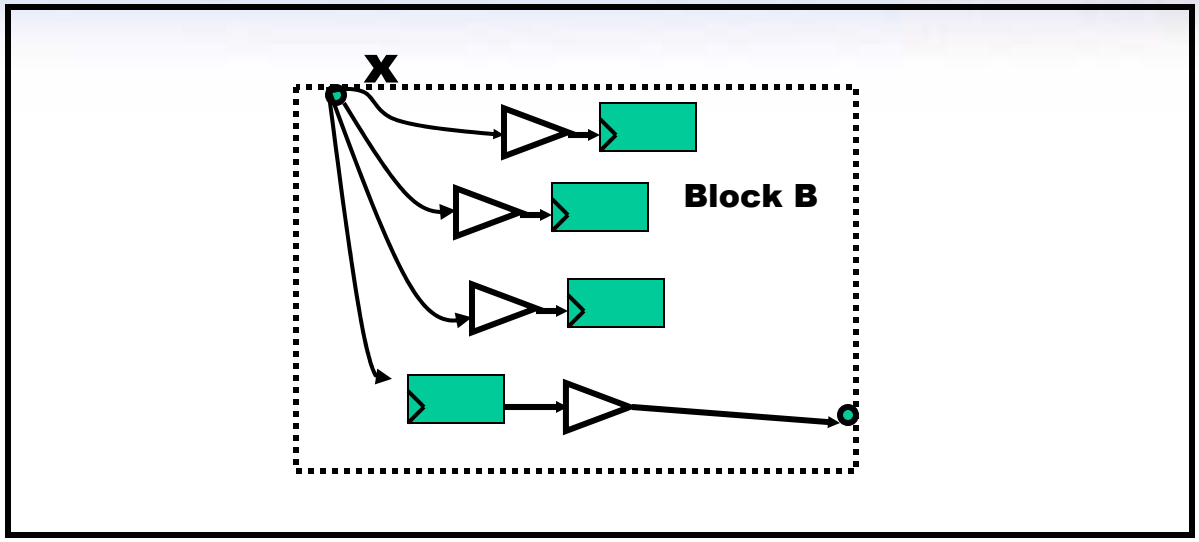
Hierarchy Flattening




```
Set_false_path -from P -to A/inv1/a
Set_false_path -from P -to A/inv2/a
Set_false_path -from P -to A/inv5/a
Set_false_path -from P -to A/ff1/d
```

Transformations: Grouping

- May have been written in isolation on a block
  - Must now be “transformed” up the hierarchy




```
Set_false_path –from x –to inv1/a
```

Becomes

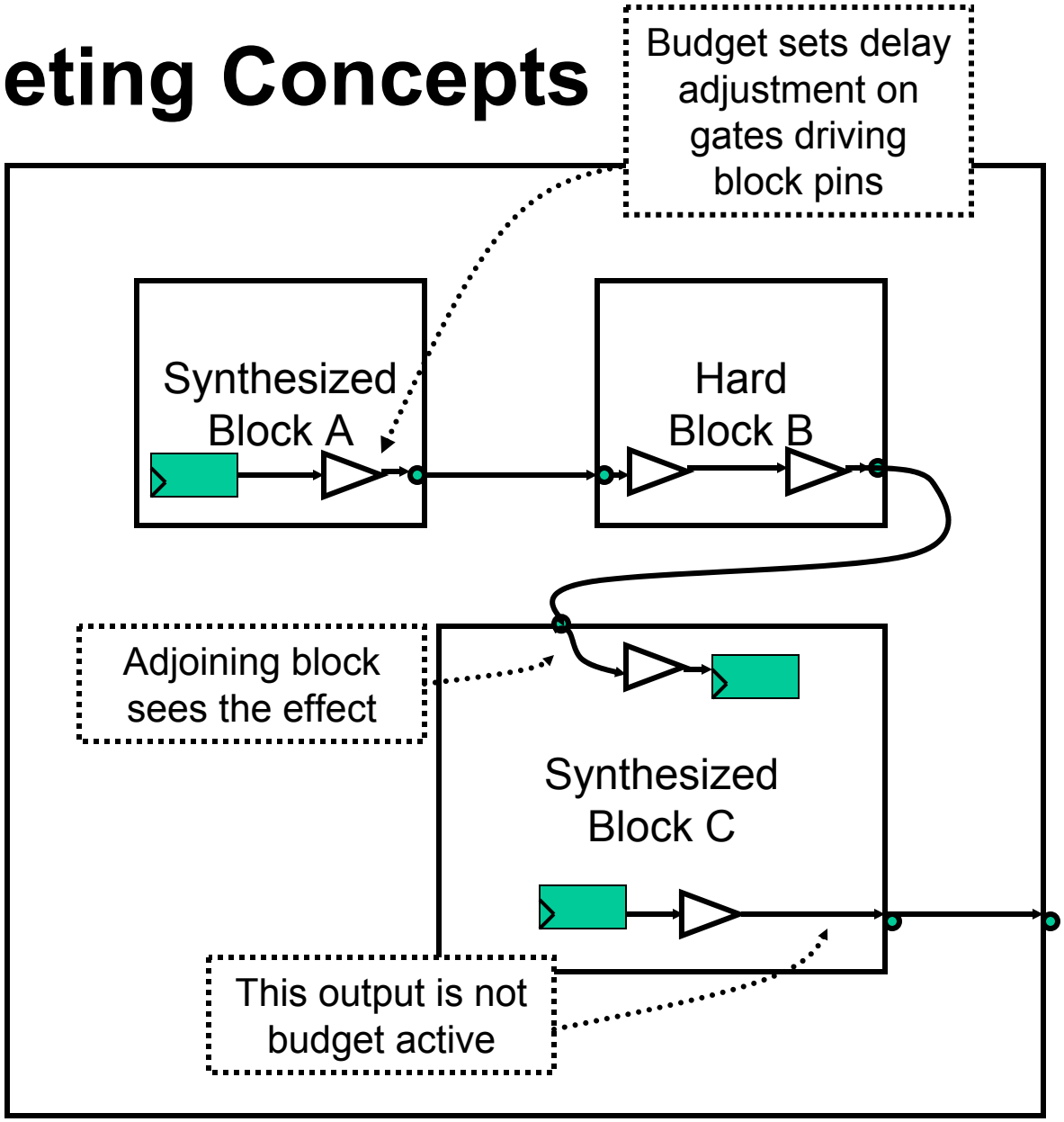
```
Set_false_path –from B/x –to B/inv1/a
```


Transformations: Budgeting

SYNOPSYS

- **Time budgeting = timing adjustments + characterizing context**
- **Timing adjustments is *interesting* algorithmic process for distributing slack**
- **Characterizing context is *complicated, detailed process* of manipulating timing constraints**
  - **Context of a port is timing "seen" by that port**
  - **Constraints inside block need to reflect context too.**

Budgeting Concepts



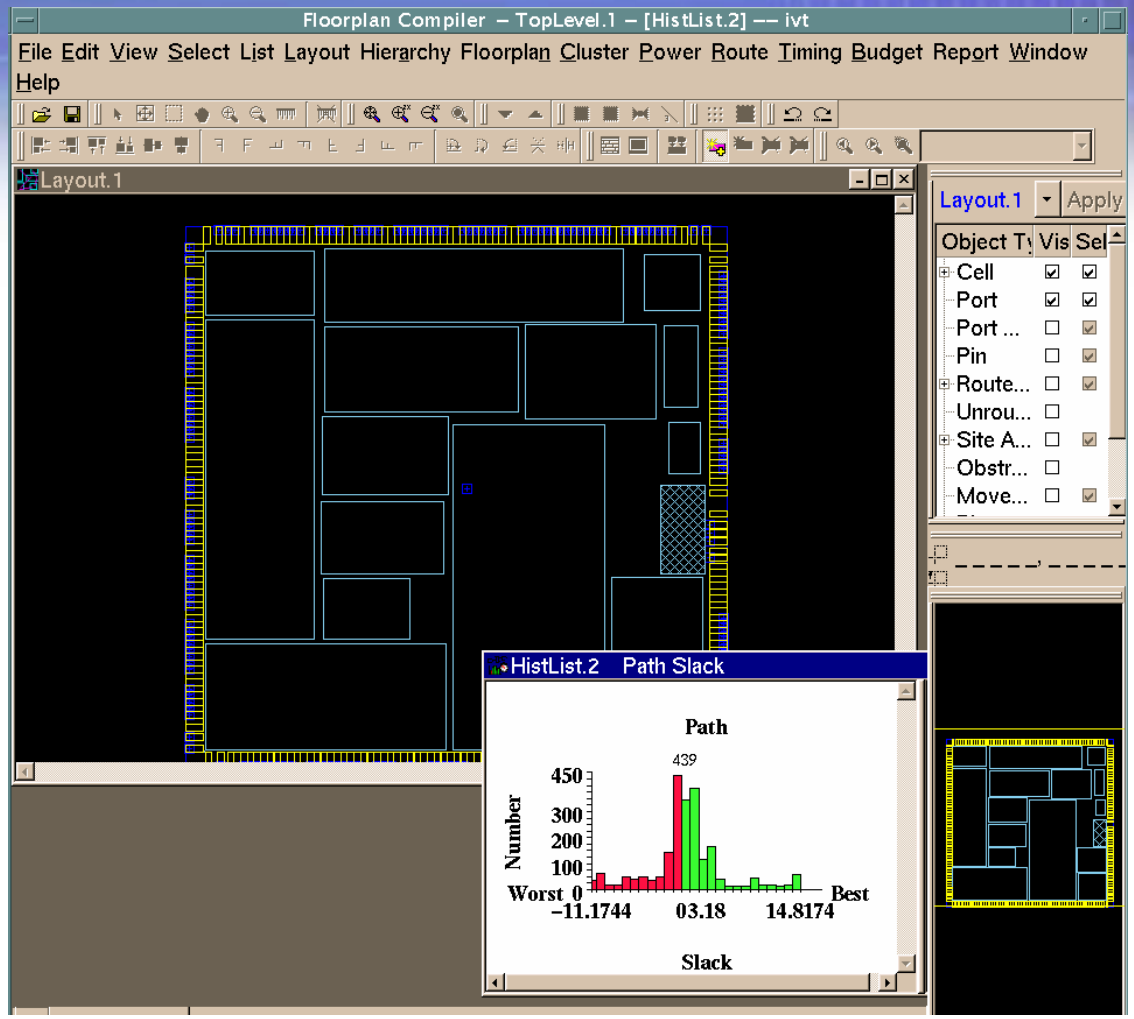
Budgeting Flow: Input

SYNOPSYS

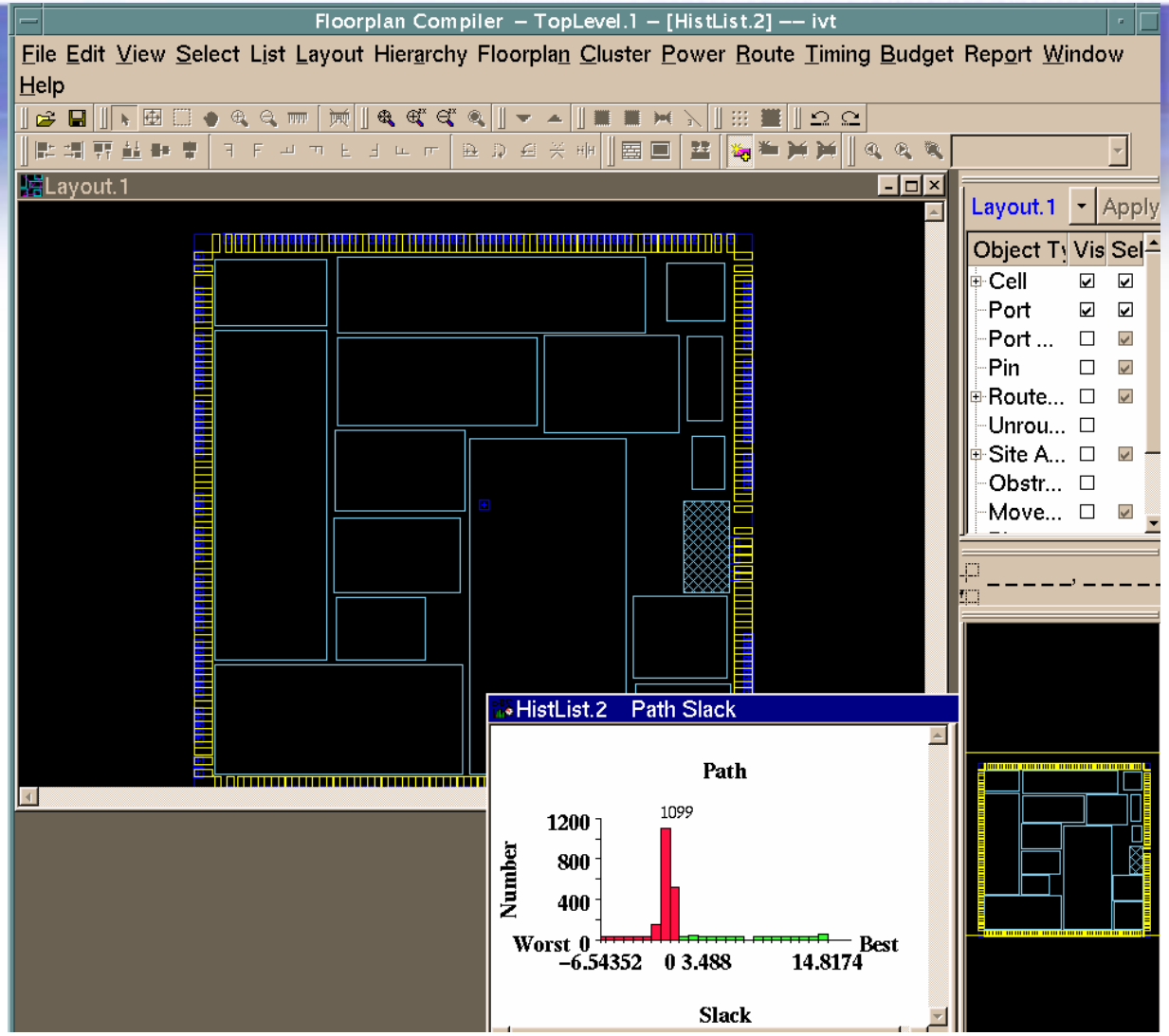
- A large chip with block structure.
  - May include blocks that are black-box or STAMP
  - Chip-level timing constraints

- Budgeting Flow: Output
  - Output of time budgeting is a directory of timing constraint files
    - one per block
    - control subsequent runs of physical synthesis subsystem

Timing before Budgeting



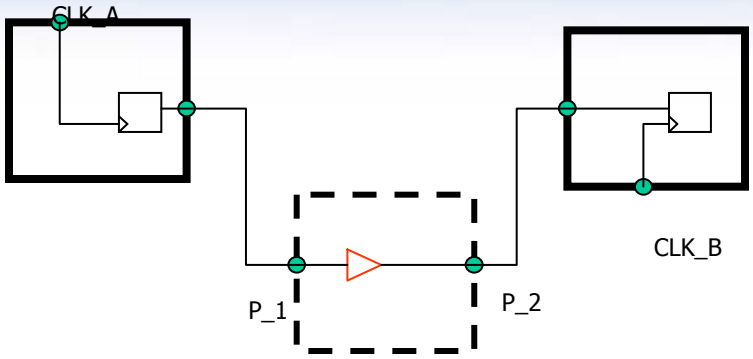
Timing after Budgeting



Characterizing the Context

SYNOPSYS

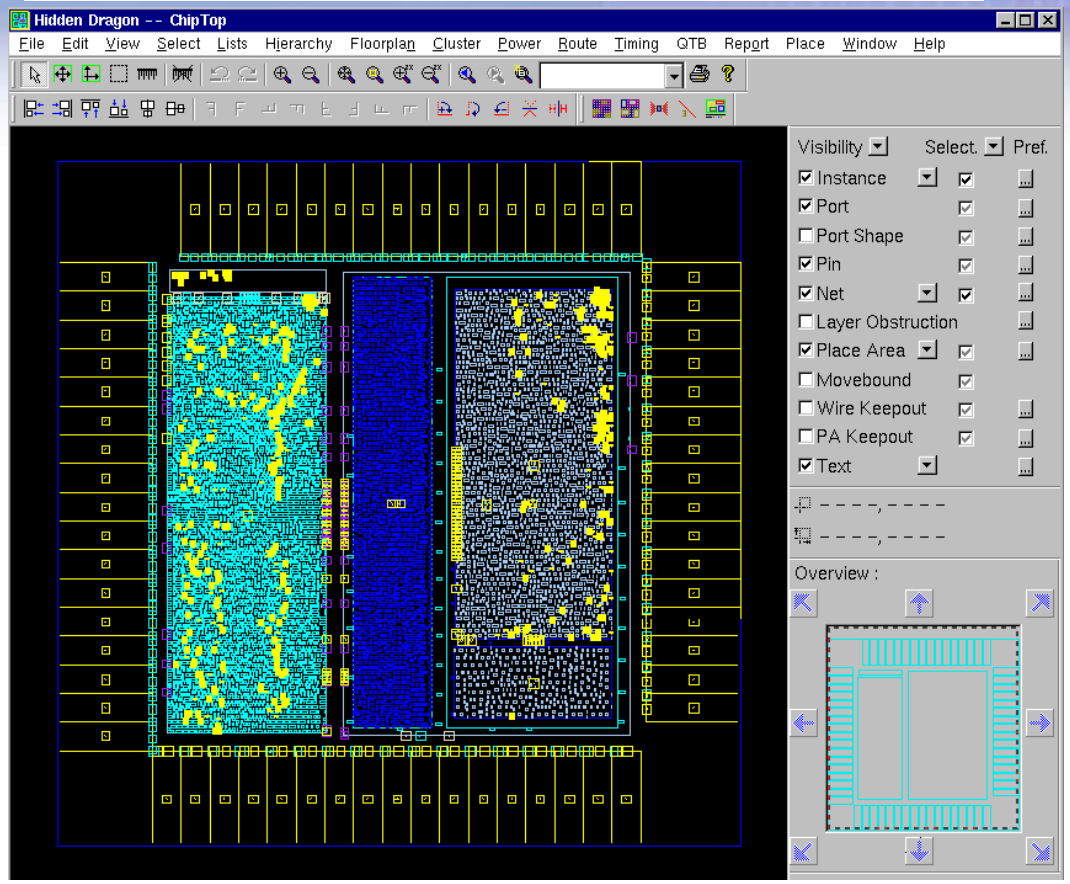
Sometimes necessary to create “virtual” clocks




```
create_clock -name clka_virtual01 -period 7.0;
create_clock -name clkb_virtual01 -period 8.0;
set_input_delay -clock clka_virtual01 3.25 [get_ports p1]
set_output_delay -clock clkb_virtual01 4.75 [get_ports p2]
```

Timing Abstraction: Processing SDC

Must identify border circuitry and only builds timing model for active circuitry. In this picture, yellow cells are not active.



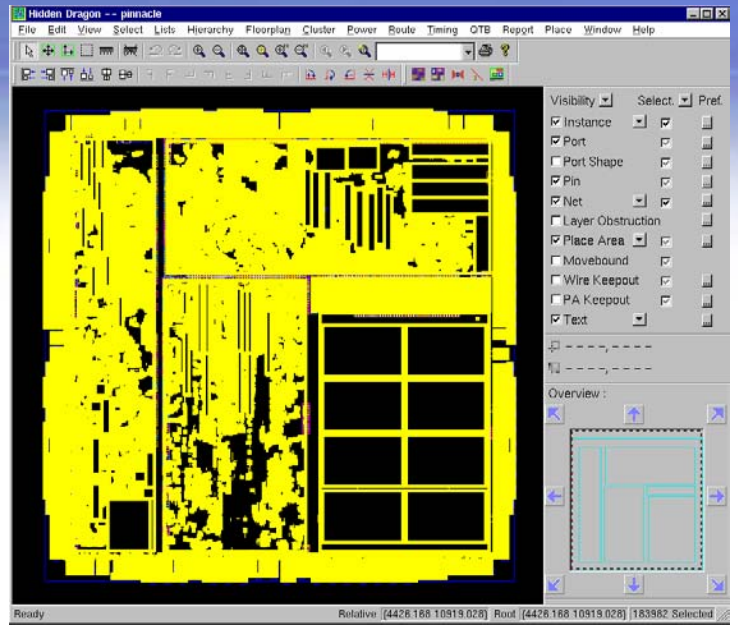
Timing Constraint Processing for Abstraction

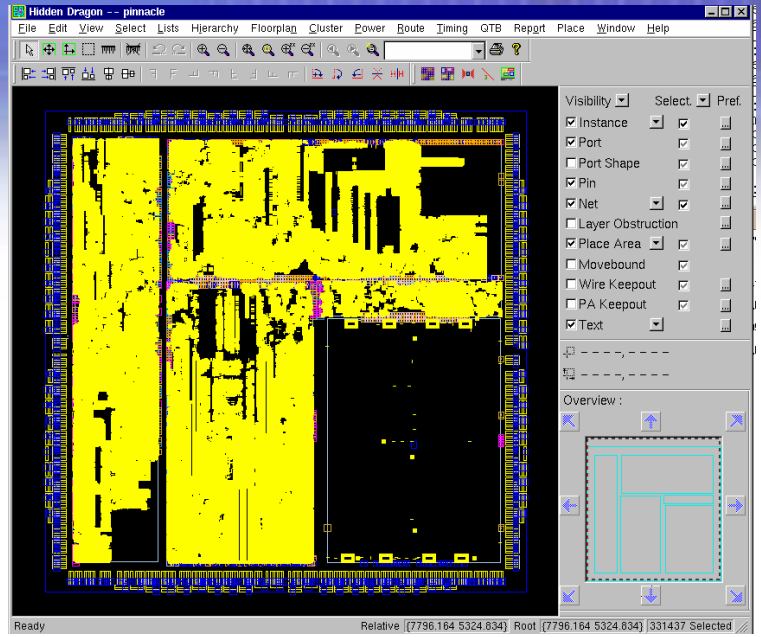
SYNOPSYS

- Necessary to “understand” constraints in order to build valid abstraction
  - Constraints may refer to points deep inside blocks: these must be preserved
- Processing general Tcl based SDC can be tricky

TAU Overt Cells (184K) TAU Covert Cells(331K)

515K cells total





ECO's and other Issues

SYNOPSYS

- Changes in timing constraints can come at any point
  - User wants to make them in original format
  - User wants to see constraints in compact format, not expansion
  - Mangling of names, expansion, can lead to frustration and error.
- Timing Constraint processing is NOT simplified in other technologies
  - FPGA's / one-mask processes, etc do not have simplified constraints
  - SOC / advanced power management complicate the process

Overview

- Intro and Background
- What Are Timing Constraints?
- Formats for Timing Constraints
- Hierarchy Mapping
- Time Budgeting
- Eco and other technology issues
- **Summary**

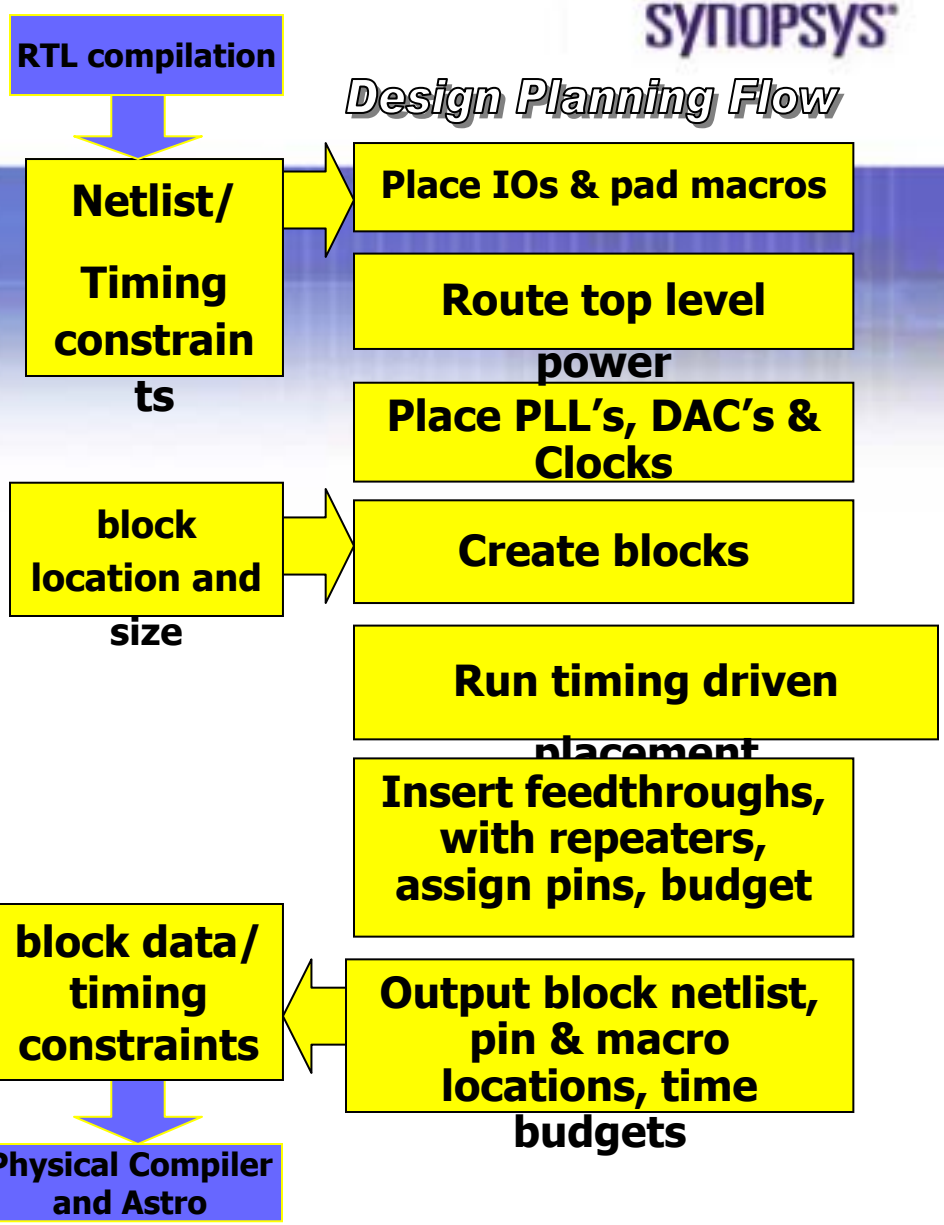
Summary

SYNOPSYS

- Complexity of constraints
  - Initially small
  - After manipulation/expansion: about 1 line per placeable object
- Tempting to “automate” this into db
  - But systems that loose design intent tend to be inefficient
  - Need to support changes in original format
- Probably best to accept original format throughout -> this has direct consequences on data model

SYNOPSYS

Design Planning Flow



At each phase, the system must be able to process the timing constraints in the original logical format