Using Tcl/CCI/Collections to turn EDA Cousins into Sisters

EDP Symposium
April 26, 2004

V1.0

Outline

- **What is the problem?**
- **What is the solution?**
- **Background on Tcl/CCI**
- **Integration of Multiple Tools using Tcl**
- **Experience / Performance**
- **Summary**

Problem

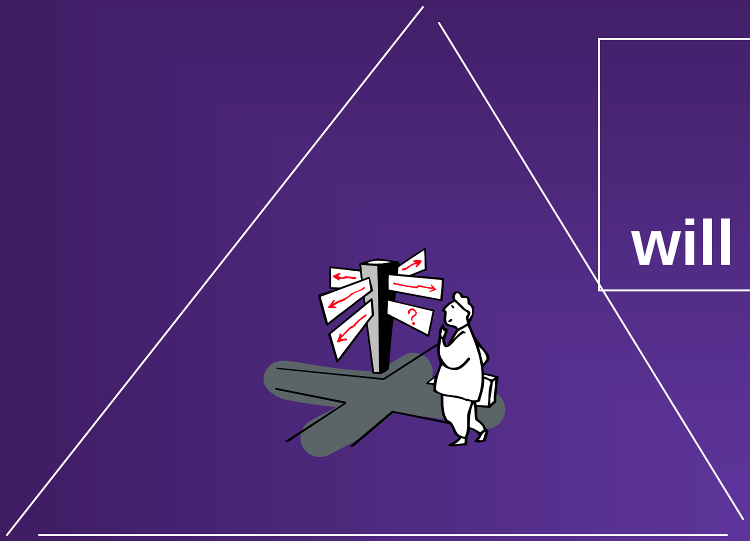
- The RTL to GDSII flow involves multiple
  - Steps
  - Abstractions
  - Tools
  - Data representations
- Users & tools see interfaces that are
  - Inconsistent
  - Redundant
  - Inefficient

Interfaces must recognize conflicting objectives

Each group wants observability and control…

Designer +
In House Support

But ultimately, the designer
will determine the winner



Vendor
DataModel
group

Standards
Committees

Previous Approaches

- EDIF
- Frameworks
- CHDStd
- Bridges
- …

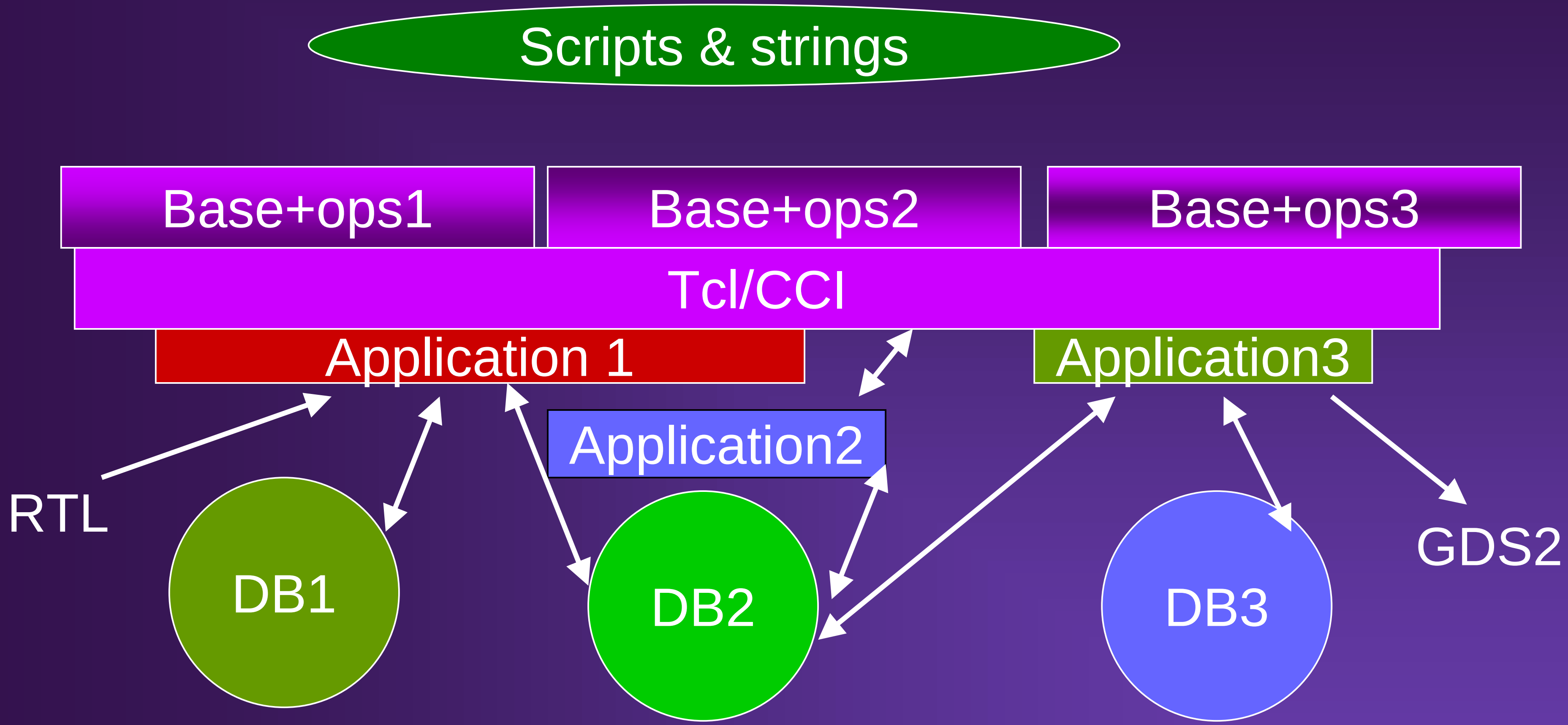
Outline

- What is the problem?
- What is the solution?
- Background on Tcl/CCI
- Integration of Multiple Tools using Tcl
- Experience / Performance
- Summary

Our approach: common (User) interface

- Users do not care what is on the other side of the interface
- Standardize on the interface, not the database
- Allow (and encourage) multiple representations to serve multiple needs.
- Since the primary interface for most users is the script – concentrate on that.

Tcl/CCI as an Overarching Interface



Basic interface components

- Interface is comprised of Tcl plus
  - *Collections* for efficiently representing large groups of objects
  - *CCI* for consistent command syntax
  - *Attributes* for database queries
- This interface is then ported onto multiple data representations and into multiple applications

Outline

- What is the problem?
- What is the solution?
- Background on Tcl/CCI
- Integration of Multiple Tools using Tcl
- Experience / Performance
- Summary

Tcl Basics

- Tcl : *Tool Control Language*,
  - Developed by John Ousterhout, UC Berkeley.
  - Widely used for scripting and GUI
- CCI : Common Command Interpreter,
  - An extension layer on top of Tcl.
  - Support collections
- Tcl/CCI : the Synopsys standard language,
  - Used by PT, DC, PC.
  - SDC is a dialect of Tcl/CCI.

CCI

- CCI: *Common Command Intepreter*, a layer on top of Tcl.
- CCI provides consistent, easy to use syntax for commands.
  - Set_input_delay –clock c 3.2 clock1
  - Set_input_d 3.2 –clo c [get_clocks *1]
- CCI also provides
  - Help, man, history, !reuse, other stuff…

Collections: not just a hobby

- Power = Force * Speed
- "Power" of User Interface =
  (ability to pull out objects & move them) * speed
  - Collections provide many facilities to select just the objects you want and put them into groups.
  - Collection "handles" allow you to store them efficiently
  - These can then be passed to any operation
  - Selection/access process takes very little time
- This makes each command much more effective than it would be if written independently

Collection Design objects

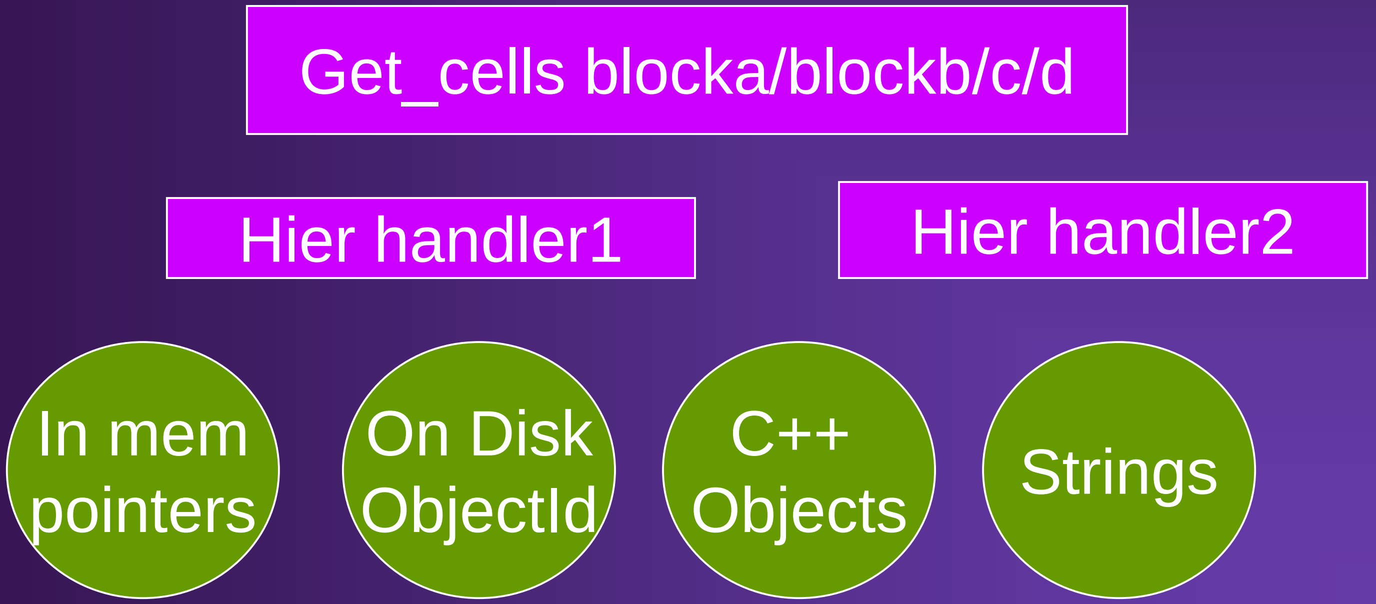
- A 1st class object is one that
  - has a name
  - has attributes (accessible with “get_attribute”)
  - may appear in collections
  - need not be persistent
- Objects may be from the design (e.g. “cell”), from the library (e.g. “lib_cell”), or only present at runtime (e.g. “clock”, or “path”).
- Attributes are not just inherited from the DB. Most are computed dynamically, e.g. “area”
- CCI Objects are not name strings. They define the type of the object, and are automatically managed as the design is updated.

Collections

- Collections are internal data structures representing ordered lists of first class objects.
- Sample operations on collections
  - set v [get_port “fred”]
  - query_objects [get_nets *]
  - set h
    [get_cells -filter “is_hierarchical == TRUE” *]
  - set big_first [sort_collection $h area]

Porting Collections to Multiple tools

- **Because collections are a high level concept, they can be implemented in radically different ways on multiple tools**



Get_cells blocka/blockb/c/d

Hier handler1

Hier handler2

In mem pointers

On Disk ObjectId

C++ Objects

Strings

Examples of Basic Collection Operations

- Set p [Get_ports –filter “direction==input” *n*]
  {n1 n2 in1 in2 in3 bnnn}
- Set i [sort_collection $p name]
   {bnnn in1 in2 in3 n1 n2}

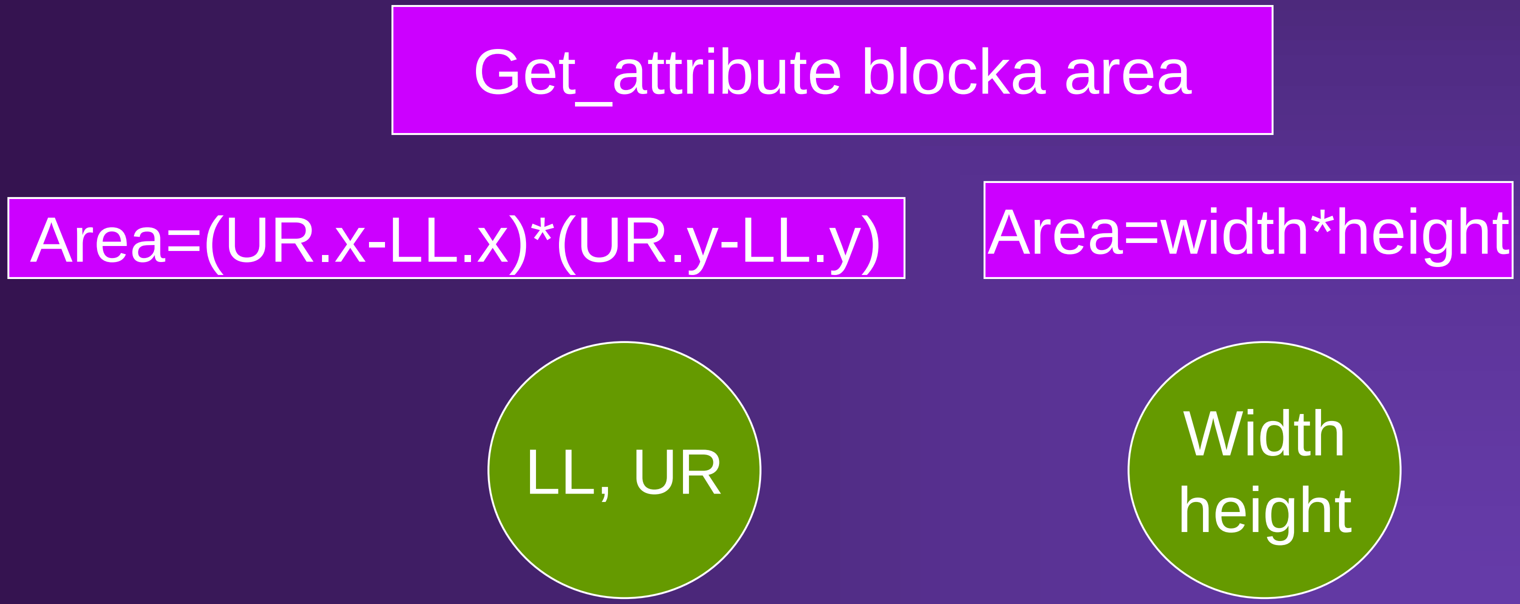
-
  ```
  foreach_in_collection nn $i {
      echo [get_att $nn full_name] \
        [get_att $nn area]
   }
  ```

Attributes

- Attributes are central to Tcl/CCI, providing a major amount of expression and capability
- Astro/CCI provides many core attributes for objects
  - Core: full_name, object_type, bbox, ref_name, etc.
  - Timing: max_fall_slack, min_rise_slack, etc.
  - Milkyway: Object_id, attached_files
- Using attributes it is easy to write convenient report commands (*e.g.* list of pins with large neg. slacks).

Porting Attributes

- **Attributes can be retrieved directly or computed, and reformatted for consistently**



Get_attribute blocka area

Area=(UR.x-LL.x)*(UR.y-LL.y)

Area=width*height

LL, UR

Width
height

Tcl/CCI and the GUI

- Tcl/tk has long been the basis of GUI's
- With collections/attributes, the GUI generates and evaluates Tcl commands behind the scene to get or modify the design
- No need to rebuild the GUI when the data model changes.
- Debugging/ testing is simplified
- Change_selection, get_selection fit naturally into collection mechanism:
  e.g. to highlight objects on the GUI window
  change_selection [get_cells –filter number_of_pins > 4" *]

Outline

- What is the problem?
- What is the solution?
- Background on Tcl/CCI
- Integration of Multiple Tools using Tcl
- Experience / Performance
- Summary

Why use a Tcl interface?

- More compact API specifications than a Framework
  - C, (and especially, C++) requires lengthy API spec
  - Even “object oriented” interfaces reveal internal details in C++ (“There is no privacy – get used to it”)
  - C/C++ requires memory management. This might conflict with the memory management of the application

Tcl as an SQL…

- Provides some database queries automatically
  - Get_cells –of_object [get_nets “fred”]
  - Get_cells –filter “area > $ma” blockb/*ff
  - Set non_clocks
    [get_pins [remove_from_collection
     [all_inputs] [all_clocks]]
  - Sort_collection $non_clocks slack

Object Categories

- Some objects are common to all tools in the suite
  - such as nets, ports, etc.
- Some are local to one tool (not present in any common database),
  - such as slots in a slot-filling step.
- Some are common to two or more tools in the suite communicating through databases or translators, but not necessarily understood by all other tools.
  - E.g. physical implementation tools, such as routers and DRC engines will need "wire" objects, but logic simulation tools will generally not need them.

Outline

- What is the problem?
- What is the solution?
- Background on Tcl/CCI
- Integration of Multiple Tools using Tcl
- Experience / Performance
- Summary

Most important things are not free

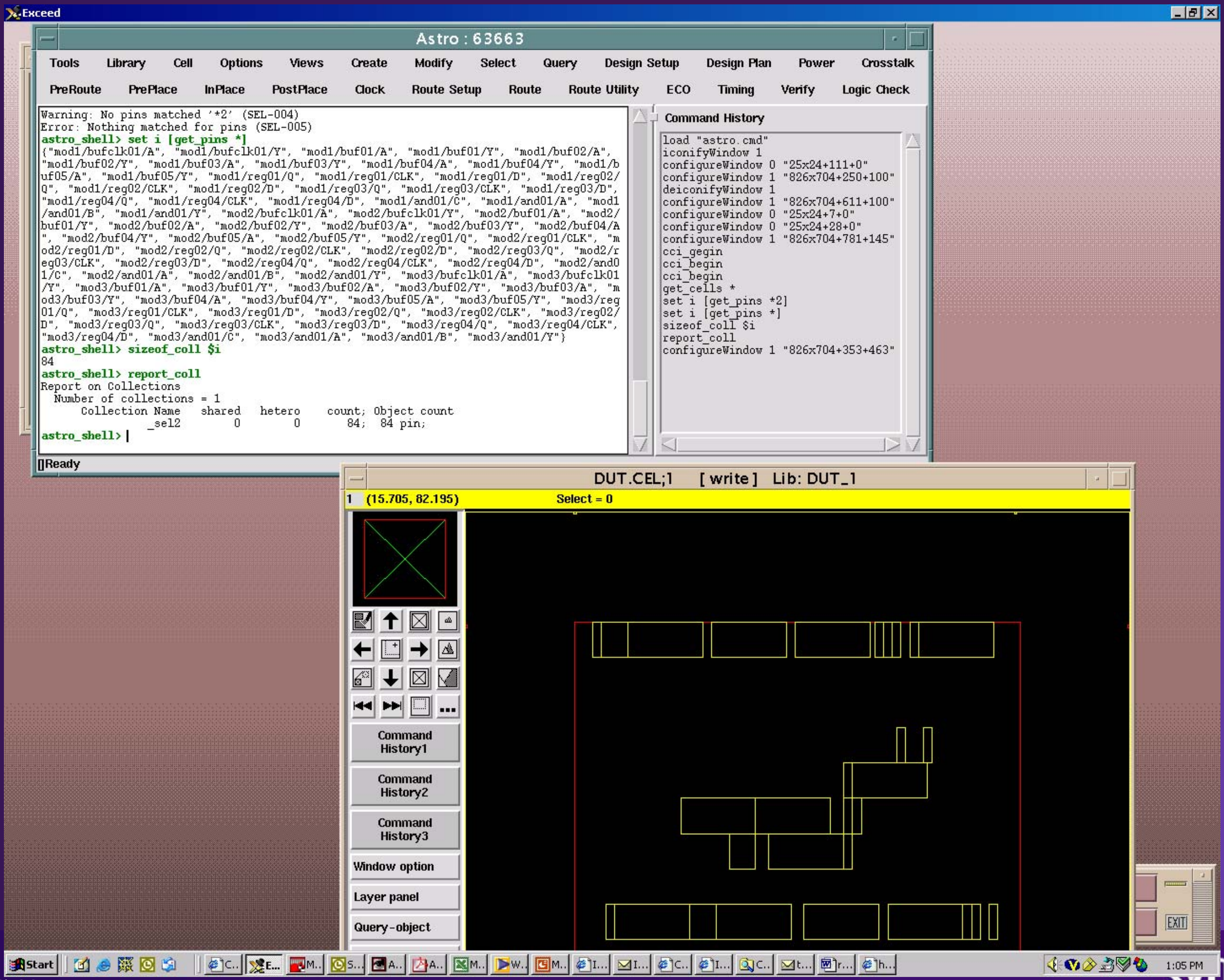
- The second most important thing in architecting interfaces is that the same concept have the same name in all tools that use it.
- The Most Important Factor is that different concepts have different names in all tools.

A not-very-important factor is having the same representation of an object throughout.

Experience with Tcl/CCI/Collections

- AE’s, R&D and designers have written elaborate operations using Tcl/CCI/collections:
  - device sizing
  - floorplanning/placement
  - area IO support, etc.
- Typically these use combination of collection “get_*”, filtered, sorted, selected, and then operated on.

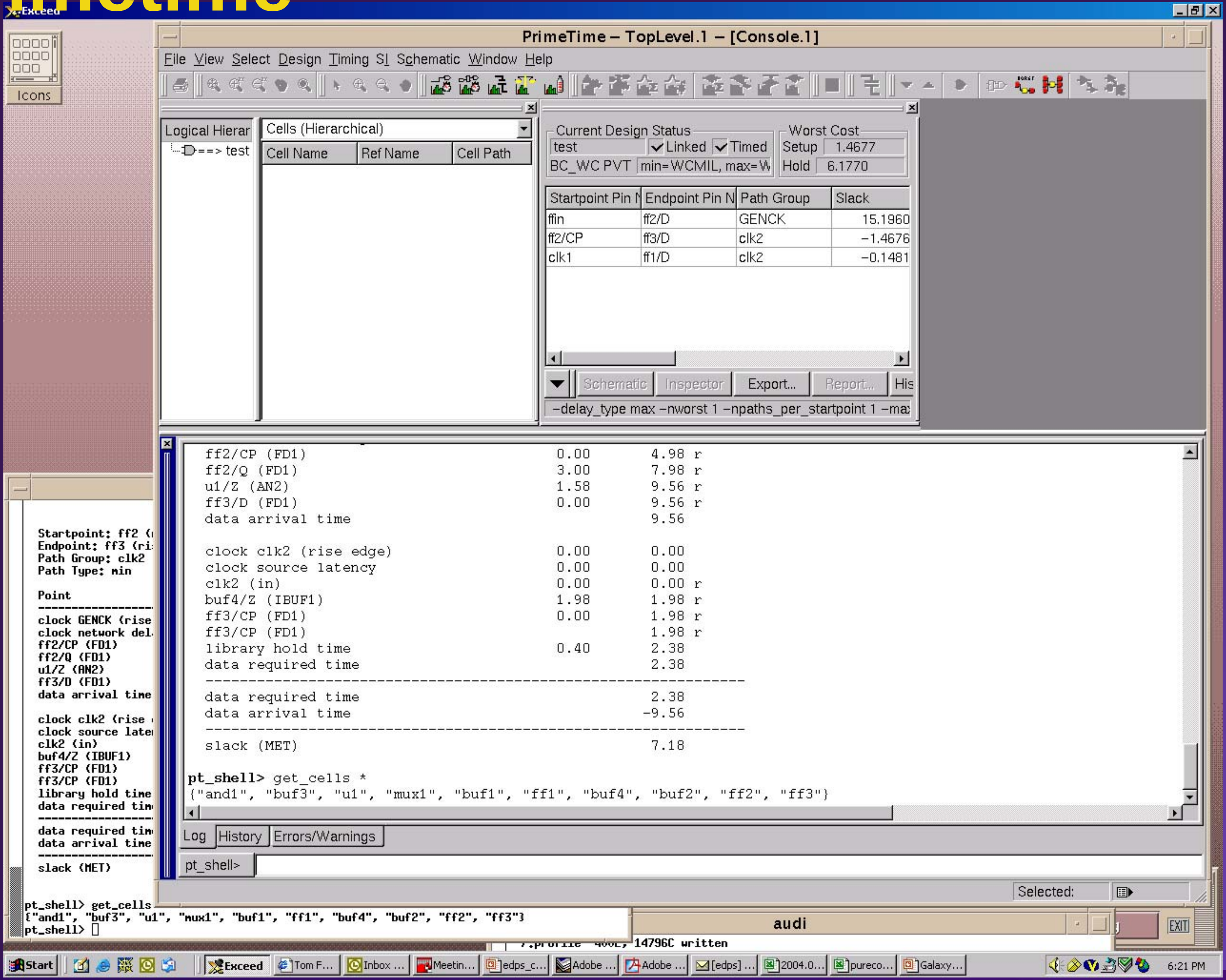
Screen Shots: Astro



```
Warning: No pins matched '*2' (SEL-004)
Error: Nothing matched for pins (SEL-005)
astro_shell> set i [get_pins *]
{"mod1/bufclk01/A", "mod1/bufclk01/Y", "mod1/buf01/A", "mod1/buf01/Y", "mod1/buf02/A",
"mod1/buf02/Y", "mod1/buf03/A", "mod1/buf03/Y", "mod1/buf04/A", "mod1/buf04/Y", "mod1/b
uf05/A", "mod1/buf05/Y", "mod1/reg01/Q", "mod1/reg01/CLK", "mod1/reg01/D", "mod1/reg02/
Q", "mod1/reg02/CLK", "mod1/reg02/D", "mod1/reg03/Q", "mod1/reg03/CLK", "mod1/reg03/D",
"mod1/reg04/Q", "mod1/reg04/CLK", "mod1/reg04/D", "mod1/and01/C", "mod1/and01/A", "mod1
/and01/B", "mod1/and01/Y", "mod2/bufclk01/A", "mod2/bufclk01/Y", "mod2/buf01/A", "mod2/
buf01/Y", "mod2/buf02/A", "mod2/buf02/Y", "mod2/buf03/A", "mod2/buf03/Y", "mod2/buf04/A
", "mod2/buf04/Y", "mod2/buf05/A", "mod2/buf05/Y", "mod2/reg01/Q", "mod2/reg01/CLK", "m
od2/reg01/D", "mod2/reg02/Q", "mod2/reg02/CLK", "mod2/reg02/D", "mod2/reg03/Q", "mod2/r
eg03/CLK", "mod2/reg03/D", "mod2/reg04/Q", "mod2/reg04/CLK", "mod2/reg04/D", "mod2/and0
1/C", "mod2/and01/A", "mod2/and01/B", "mod2/and01/Y", "mod3/bufclk01/A", "mod3/bufclk01
/Y", "mod3/buf01/A", "mod3/buf01/Y", "mod3/buf02/A", "mod3/buf02/Y", "mod3/buf03/A", "m
od3/buf03/Y", "mod3/buf04/A", "mod3/buf04/Y", "mod3/buf05/A", "mod3/buf05/Y", "mod3/reg
01/Q", "mod3/reg01/CLK", "mod3/reg01/D", "mod3/reg02/Q", "mod3/reg02/CLK", "mod3/reg02/
D", "mod3/reg03/Q", "mod3/reg03/CLK", "mod3/reg03/D", "mod3/reg04/Q", "mod3/reg04/CLK",
"mod3/reg04/D", "mod3/and01/C", "mod3/and01/A", "mod3/and01/B", "mod3/and01/Y"}
astro_shell> sizeof_coll $i
84
astro_shell> report_coll
Report on Collections
  Number of collections = 1
      Collection Name   shared   hetero    count; Object count
               _sel2         0        0       84;  84 pin;
astro_shell> |
```

```
load "astro.cmd"
iconifyWindow 1
configureWindow 0 "25x24+111+0"
configureWindow 1 "826x704+250+100"
deiconifyWindow 1
configureWindow 1 "826x704+611+100"
configureWindow 0 "25x24+7+0"
configureWindow 0 "25x24+28+0"
configureWindow 1 "826x704+781+145"
cci_gegin
cci_begin
cci_begin
get_cells *
set i [get_pins *2]
set i [get_pins *]
sizeof_coll $i
report_coll
configureWindow 1 "826x704+353+463"
```

Primetime



PrimeTime – TopLevel.1 – [Console.1]

Current Design Status: test ✓ Linked ✓ Timed
BC_WC PVT: min=WCMIL, max=W

Worst Cost: Setup 1.4677, Hold 6.1770

Startpoint Pin N	Endpoint Pin N	Path Group	Slack
ffin	ff2/D	GENCK	15.1960
ff2/CP	ff3/D	clk2	−1.4676
clk1	ff1/D	clk2	−0.1481

−delay_type max −nworst 1 −npaths_per_startpoint 1 −ma

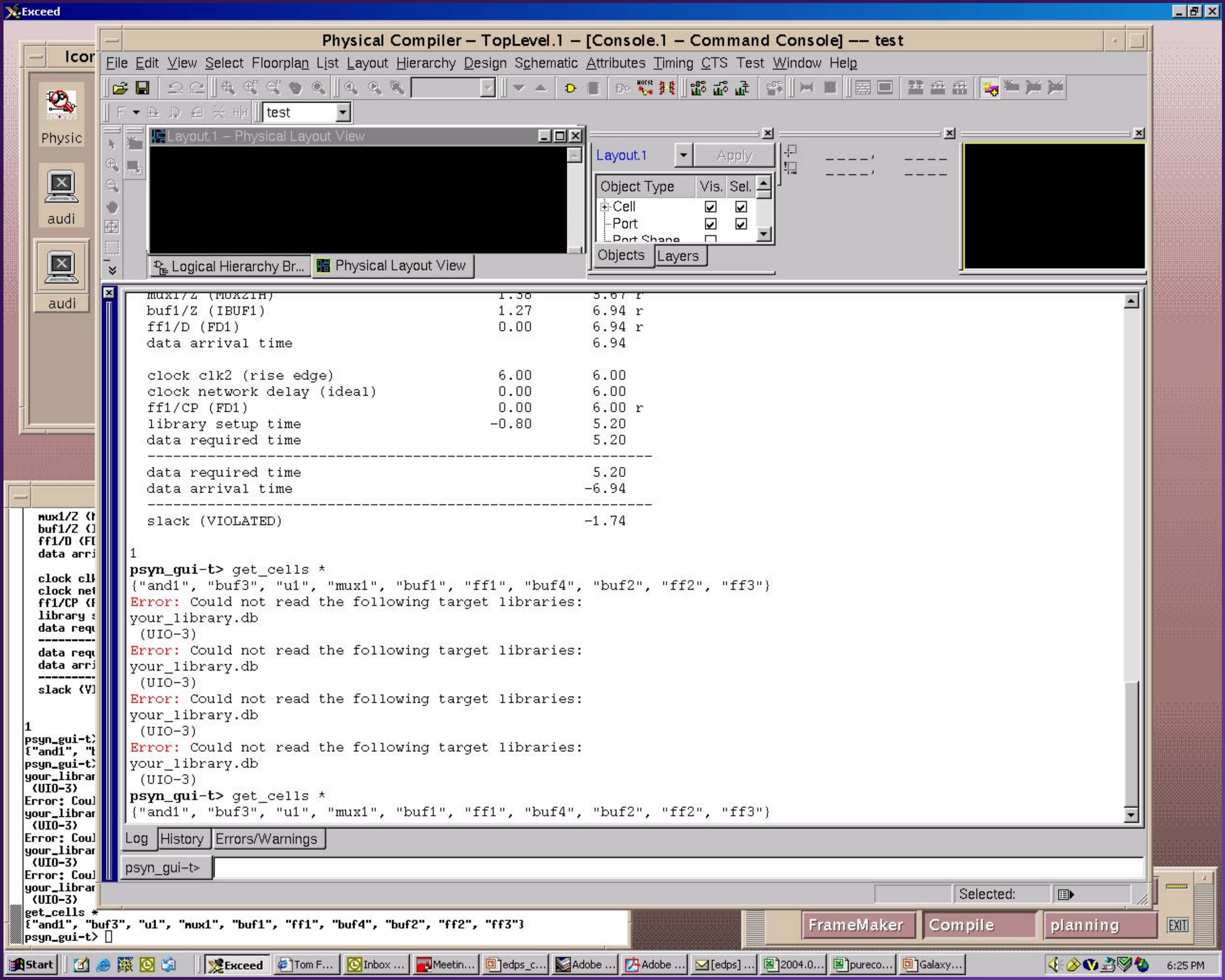
```
  ff2/CP (FD1)                              0.00      4.98 r
  ff2/Q (FD1)                               3.00      7.98 r
  u1/Z (AN2)                                1.58      9.56 r
  ff3/D (FD1)                               0.00      9.56 r
  data arrival time                                   9.56

  clock clk2 (rise edge)                    0.00      0.00
  clock source latency                      0.00      0.00
  clk2 (in)                                 0.00      0.00 r
  buf4/Z (IBUF1)                            1.98      1.98 r
  ff3/CP (FD1)                              0.00      1.98 r
  ff3/CP (FD1)                                        1.98 r
  library hold time                         0.40      2.38
  data required time                                  2.38
  -----------------------------------------------------------------
  data required time                                  2.38
  data arrival time                                  -9.56
  -----------------------------------------------------------------
  slack (MET)                                         7.18

pt_shell> get_cells *
{"and1", "buf3", "u1", "mux1", "buf1", "ff1", "buf4", "buf2", "ff2", "ff3"}
```

```
pt_shell>
```

```
  Startpoint: ff2 (r
  Endpoint: ff3 (ri
  Path Group: clk2
  Path Type: min

  Point
  ------------------
  clock GENCK (rise
  clock network del
  ff2/CP (FD1)
  ff2/Q (FD1)
  u1/Z (AN2)
  ff3/D (FD1)
  data arrival time

  clock clk2 (rise
  clock source laten
  clk2 (in)
  buf4/Z (IBUF1)
  ff3/CP (FD1)
  ff3/CP (FD1)
  library hold time
  data required tim
  ------------------
  data required tim
  data arrival time
  ------------------
  slack (MET)

pt_shell> get_cells
{"and1", "buf3", "u1", "mux1", "buf1", "ff1", "buf4", "buf2", "ff2", "ff3"}
pt_shell> □
```

```
7.profile 400L, 14796C written
```

Physical Compiler



```
  mux1/Z (MUX21H)                          1.38       5.67 r
  buf1/Z (IBUF1)                           1.27       6.94 r
  ff1/D (FD1)                              0.00       6.94 r
  data arrival time                                   6.94

  clock clk2 (rise edge)                   6.00       6.00
  clock network delay (ideal)              0.00       6.00
  ff1/CP (FD1)                             0.00       6.00 r
  library setup time                      -0.80       5.20
  data required time                                  5.20
  -----------------------------------------------------------
  data required time                                  5.20
  data arrival time                                  -6.94
  -----------------------------------------------------------
  slack (VIOLATED)                                   -1.74


1
psyn_gui-t> get_cells *
{"and1", "buf3", "u1", "mux1", "buf1", "ff1", "buf4", "buf2", "ff2", "ff3"}
Error: Could not read the following target libraries:
your_library.db
 (UIO-3)
Error: Could not read the following target libraries:
your_library.db
 (UIO-3)
Error: Could not read the following target libraries:
your_library.db
 (UIO-3)
Error: Could not read the following target libraries:
your_library.db
 (UIO-3)
psyn_gui-t> get_cells *
{"and1", "buf3", "u1", "mux1", "buf1", "ff1", "buf4", "buf2", "ff2", "ff3"}
```

```
  mux1/Z (M
  buf1/Z (I
  ff1/D (FI
  data arri

  clock clk
  clock net
  ff1/CP (F
  library s
  data requ
  ---------
  data requ
  data arri
  ---------
  slack (VI


1
psyn_gui-t>
{"and1", "b
psyn_gui-t>
your_librar
 (UIO-3)
Error: Coul
your_librar
 (UIO-3)
Error: Coul
your_librar
 (UIO-3)
Error: Coul
your_librar
 (UIO-3)
get_cells *
{"and1", "buf3", "u1", "mux1", "buf1", "ff1", "buf4", "buf2", "ff2", "ff3"}
psyn_gui-t> 
```

Tcl replaces the API for many internal operations

Application 1: budgeter

```
bdg_find_port(pp);
nd = bdgt_get_delay(pp);
Sprintf(buf, “set_input_delay [get_port %s] %f”, pp, nd);
Cci_eval(buf);
```

Application 2: timer

```
Static void tmr_set_input_delay(tmr_port p,
      float del);
```

Summary

- Many companies have script-ware as one of their most valuable assets
    - Using a common Tcl interface leverages this
    - Designs, EDA workers, CAE's all have access to it, including extending it
    - Success of SDC demonstrates the effectiveness
- Sharing an (inter) face can work wonders…