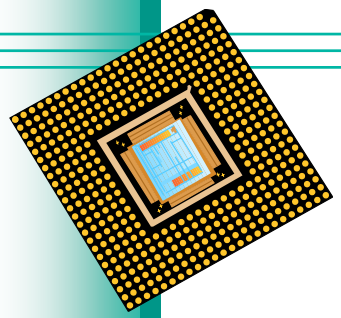


The BOA Methodology

Presentation to SNUG `97

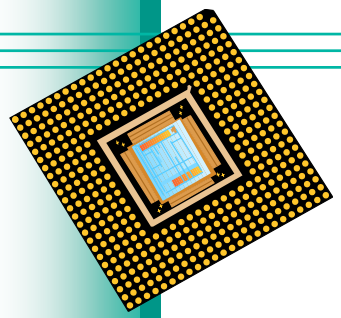
Wilson Snyder

Digital Semiconductor

ALPHA SM GENERATION

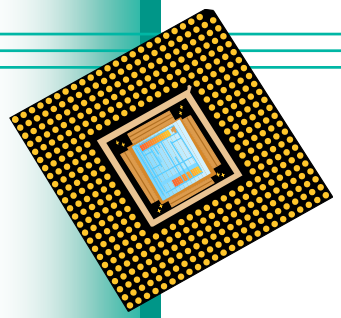
digital

PCI LOCAL BUS



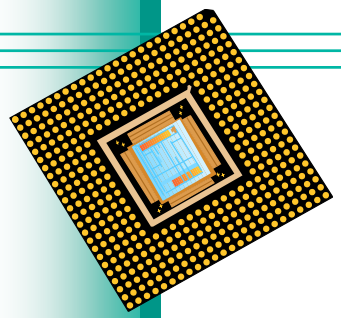
Agenda

- What is Boa?
- Goals & Overview
- Timing File
- Re-Constraining
- Synthesis script format
- Parallel compilation
- Suggestions to Synopsys
- Conclusions & Credits



What is Boa?

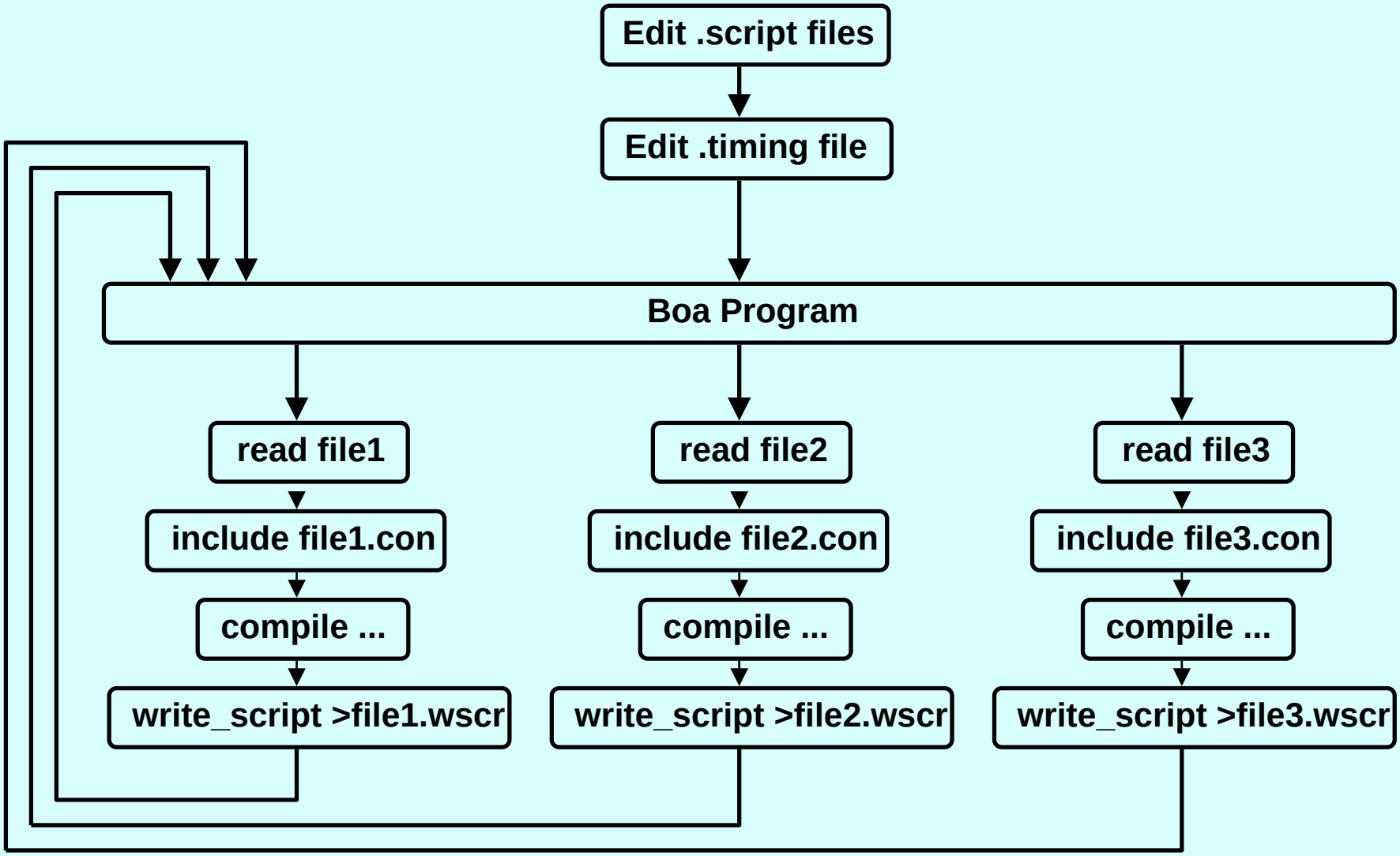
- A constraint-driven compile methodology
  - Begins with manual time-budgeting, then improves
- Estimated to half cycle time
  - Compared to top-down or bottom-up techniques
- Used on core logic chips:
  - Kodiak P 50K Gates 66 MHz
  - Kodiak C 60K Gates 66 MHz
  - 21271-C* 180K Gates 83++ MHz
  - 21271-D* 120K Gates 83++ MHz
  - 21271-P* 120K Gates 83++ MHz
- ***Marketing Plug**
  - **Chipset for the 21264, the fastest microprocessor.**
  - **Presented at Microprocessor Forum, Oct `96.**



What were Boa's goals?

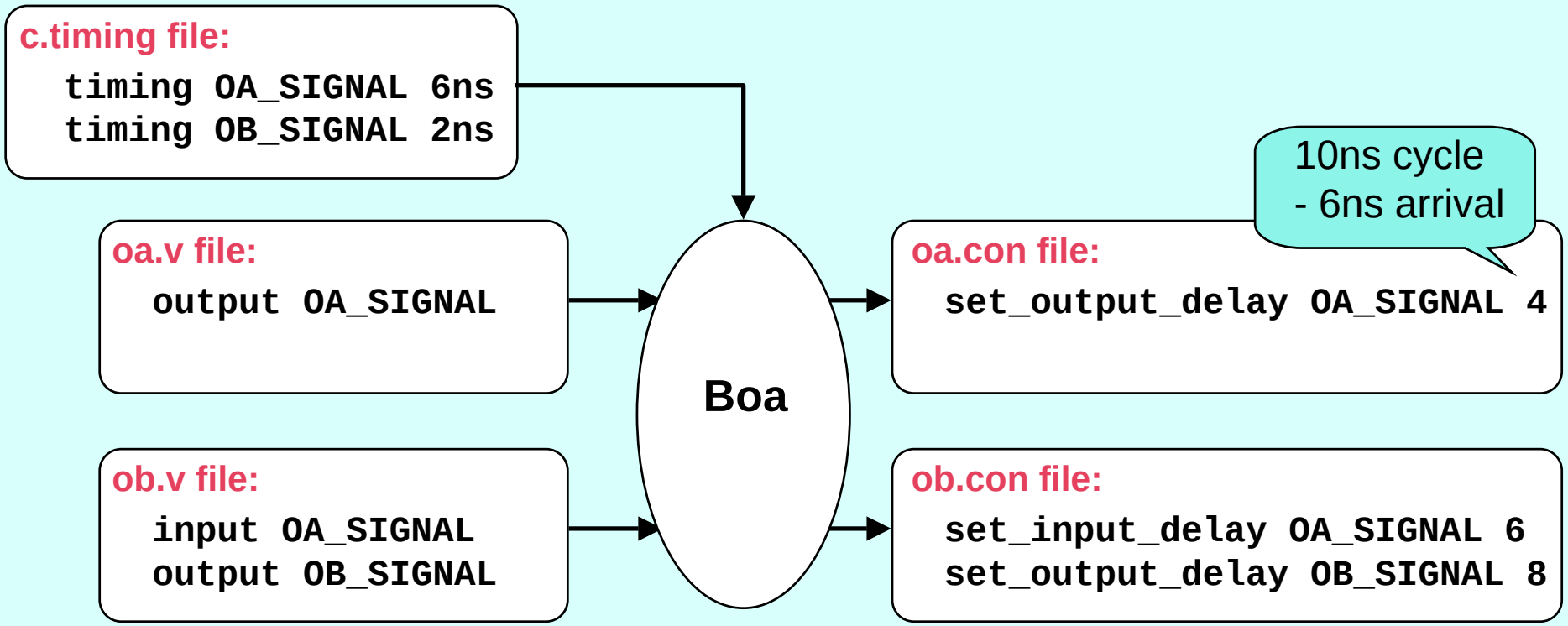
- #1 Best cycle time
  - Hand generated time budgets do well but take effort
- Eliminate redundant information
  - set_input_delay on signal destination modules
  - set_output_delay on signal source module
  - set_fake_path in Synopsys, Timing Analyzer, ...
- Allow stand-alone block compiles
  - Without creating special scripts
- Enable structural instantiations
- Hide internal details where possible

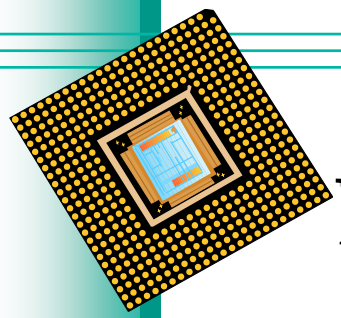
Boa Methodology Flow



Boa Timing File

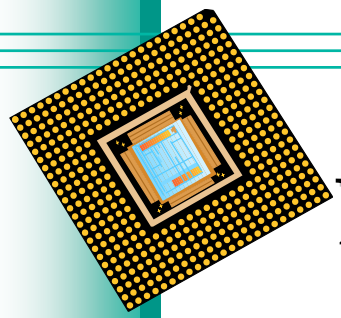
- All port constraints are stored by signal in a single file
- Users create the .timing file, and keep it up-to-date.
- Boa reads the .timing file, and creates .con constraint files for every Verilog module.





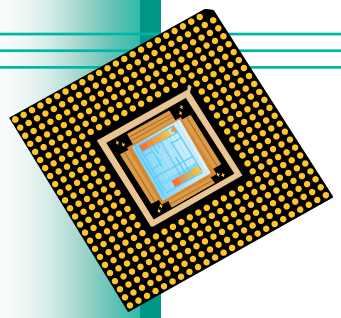
Boa Timing File Commands (1 of 2)

- timing {signal} {time}
  - Places **`set_input_delay`** on ports that input the signal.
  - Places **`set_output_delay`** on ports that output the signal.
  - Delay times are recalculated by Boa after the first loop
- weight {signal} {weight}
  - Places **`group_path`** on a single port.
  - Weights may be recalculated by Boa.



Boa Timing File Commands (2 of 2)

- loading {signal} /wire /port
  - Places **`set_load`** or **`set_wire_load`** in constraints.
  - If omitted for any signal, has reasonable default.
    - (Synopsys defaults to zero loading! Definitely optimistic!)
- driving {signal} {cell}
  - Places **`set_driving_cell`** in constraints.
  - If omitted for any signal, has reasonable default.
- path {signal}
  - Places **`set_false_path`** in constraints.
  - Passed on to downstream static timing analyzers.



Timing Feedback via Write_script

- To correct timing file errors, we want feedback of actual timing numbers.
  - Could use a static timing analyzer or **`report_timing.`**
  - **`Characterize`** and **`write_script`** do the same thing!
- Make a .wscr **`write_script`** output for every module.

oa.wscr file:

```
set_output_delay OA_SIGNAL 5
```

==

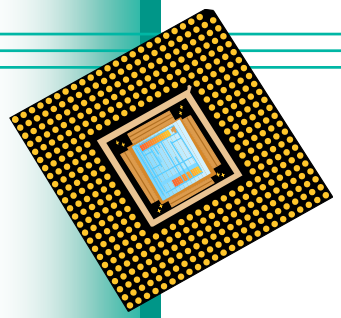
The soonest destination of the signal needs the input before (cycle_time-5ns).

ib.wscr file:

```
set_input_delay OA_SIGNAL 3
```

==

The latest source of the signal generates the output 3ns after the clock edge.



Boa Re-Constraining (1 of 2)

- Slack Paths

Take arrival and needed times

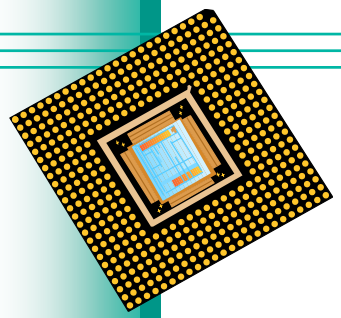
Arrival Needed

Calculate slack

Scale slack by source logic %age

Add scaled slack

Updated



Boa Re-Constraining (2 of 2)

Slack Paths

Take arrival and needed times

Arrival Needed

Calculate slack

Scale slack by source logic %age

Add scaled slack

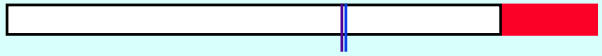
Updated

Breaking Paths

Take arrival and needed times

Needed Arrival

Calculate effective cycle time



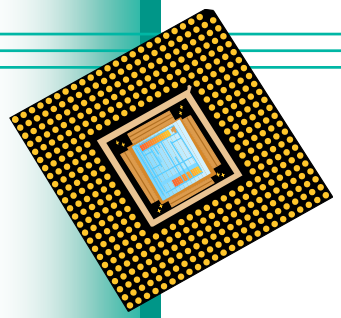
Find midpoint of needed & arrival

Scale midpoint to make cycle time

Updated

Create weight (group_path)

Weight = 1 + 6 * (violation/cycle time)



Boa Annotation

Show signal with timing in source code:

```
wire PSM_LDLADR`3.0,6.1,4.6,8.1,3.2' =
          PSM_FRAME`4.0,3.3,3.7,2.7,-1.3'
        & PSM_ENLD`2.0,3.9,2.4,7.2,4.6'
```

3.0	(Gray)	Original time user entered in .timing file
6.1	(Yellow)	Updated time calculated by Boa
4.6	(Purple)	Arrival time, from source logic delay
8.1	(Blue)	Needed time, from destination logic delay
3.2	(Green)	Slack (Needed-Arrival)
-1.3	(Red)	Violation Slack (Needed-Arrival)

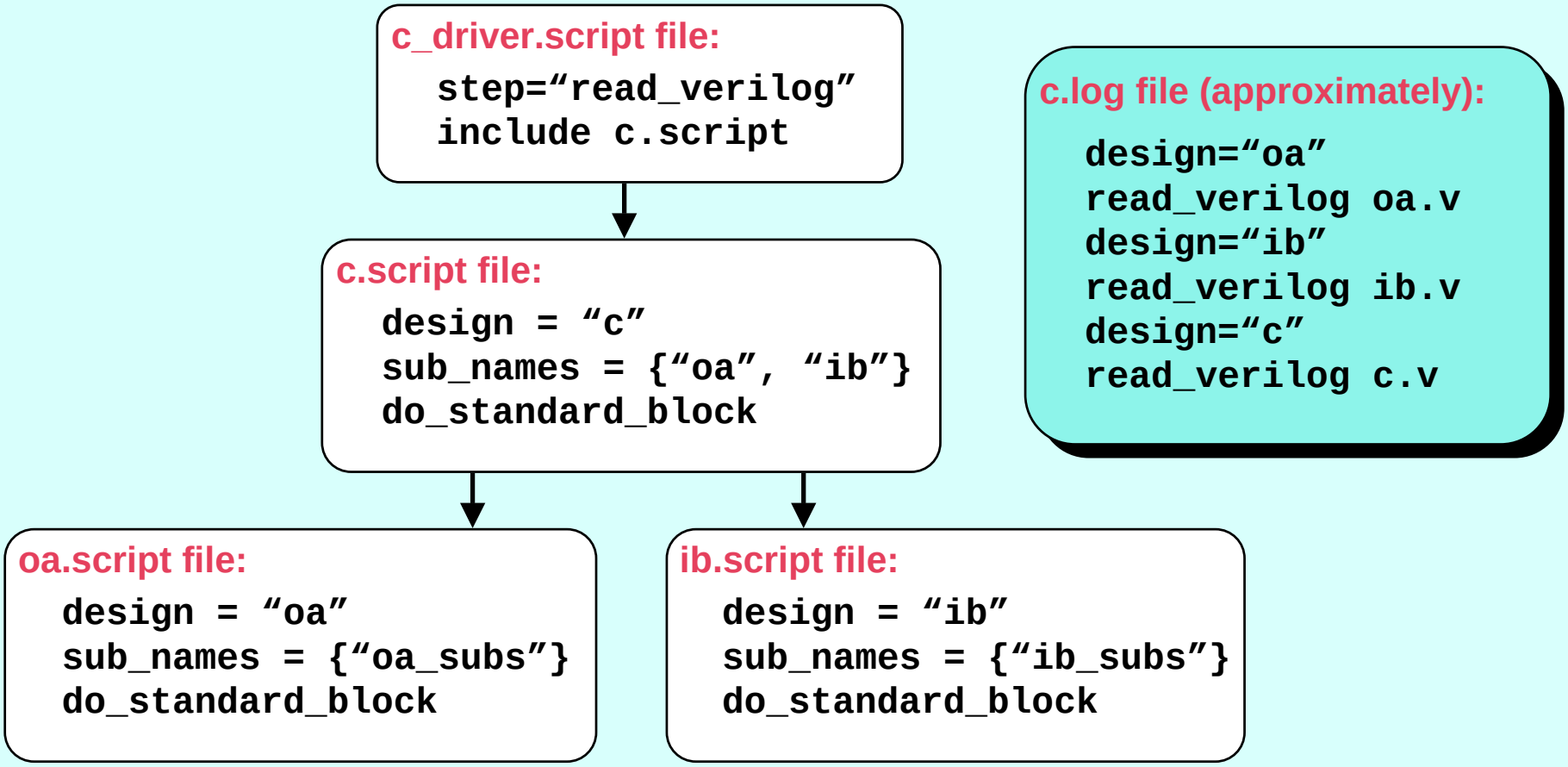
How it works:

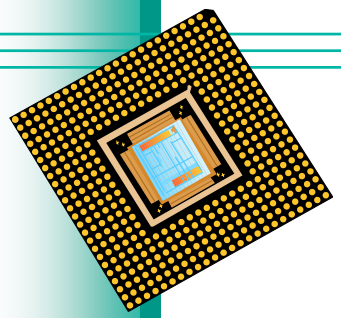
Using GNU Emacs 19:

- Read the source file
- Search and replace SIG with SIG`3.0...'
- Create overlays for each highlight color

Boa Script Flow

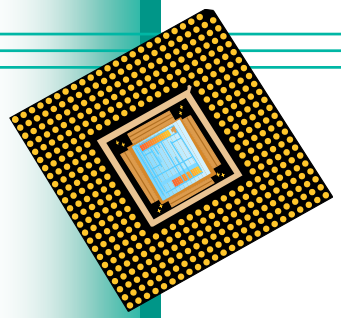
- Scripts are designed so that steps can be performed on any level of the design without changing any of the scripts.





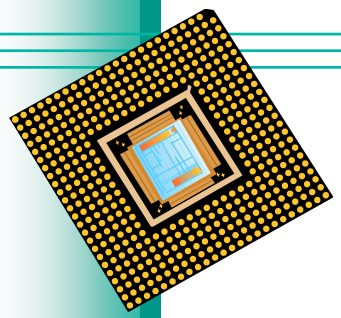
Boa Parallel Compiling

- With each block being compiled separately, we can compile each block on a different machine!
- Split script commands into tasks, indivisible blocks of commands that must execute together.
  - `task { echo “Do this on some machine.”}`
  - `primary_task { echo “Do this on machine #1.”}`
  - `non_primary_task { echo “Do this on machines #2...n.”}`
- If tasks need to be completed in order, add a sync.
  - `task { echo “Finish this task.”}`
  - `task { echo “And this task”}`
  - `sync`
  - `echo “Before printing this.”`
- Parallelization gives > 3 times the throughput when using 4 machines!
- Code included on Synopsys SNUG website.



Synopsys Improvements

- Use multiprocessors and network computes
- Associative arrays (like Perl)
- Add a “need_compile” attribute, is_mapped breaks
  - Reset on behavioral code
  - Set on a compile OR structural code
- Compile -so_high_that_youll_try_everything
  - Roughly: compile -map_effort high (2 times)
  - compile -map_effort high -incremental (2x)
- Disable echoing mode
  - foo = “x” >/dev/null
- Latch timing is broken
  - Can’t analyze both D->Q flow-through and D->clock paths



Credits

- Much thanks to the co-developers of Boa:

  Paul Wasson, **who did much of the Boa program,**
  named Boa,
  and got me into Perl!

  Steve Kolecki, **who provided ideas,**
  and adapted the methodology to the 21271

- For the full paper, search for Boa using SOLV-IT!

Please remember to fill out
the feedback forms!