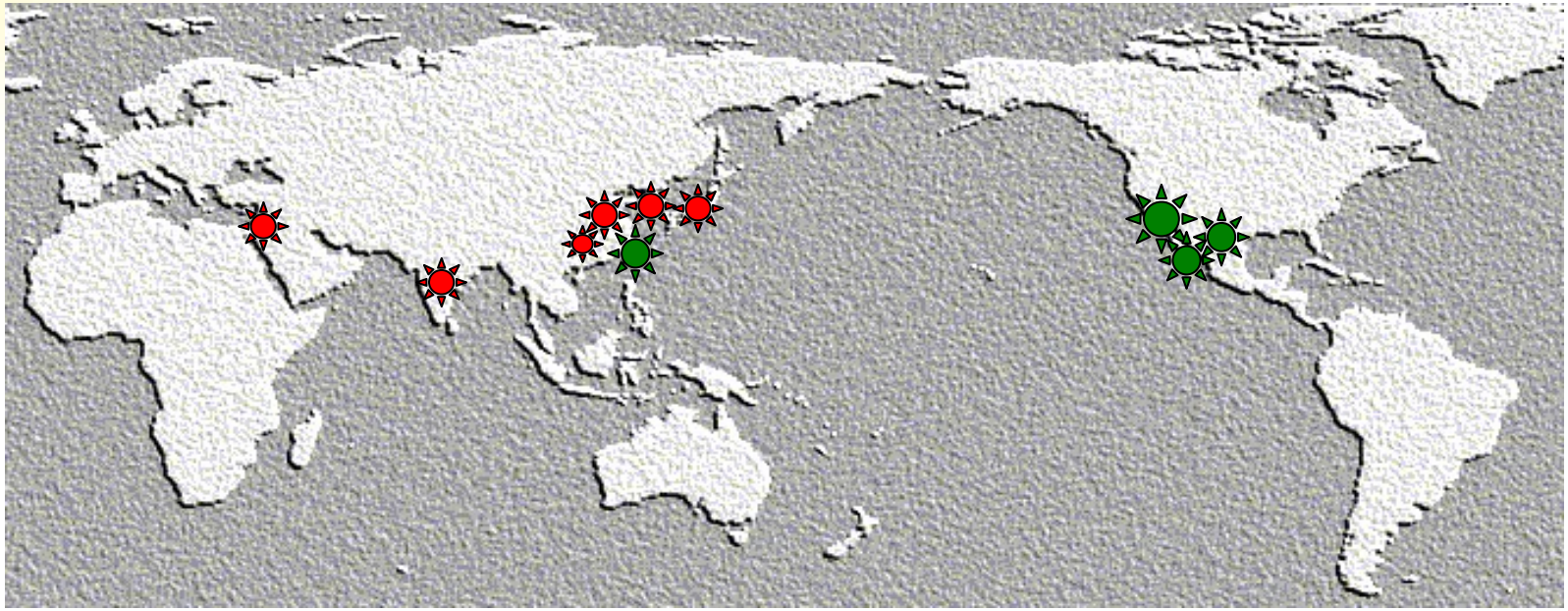
Incentia Logic/Low Power Synthesis Solution
DesignCraft

Incentia Design Systems, Inc.

October, 2010

Company Introduction

- Mission: SoC nanometer timing and synthesis technology leader
- Channels



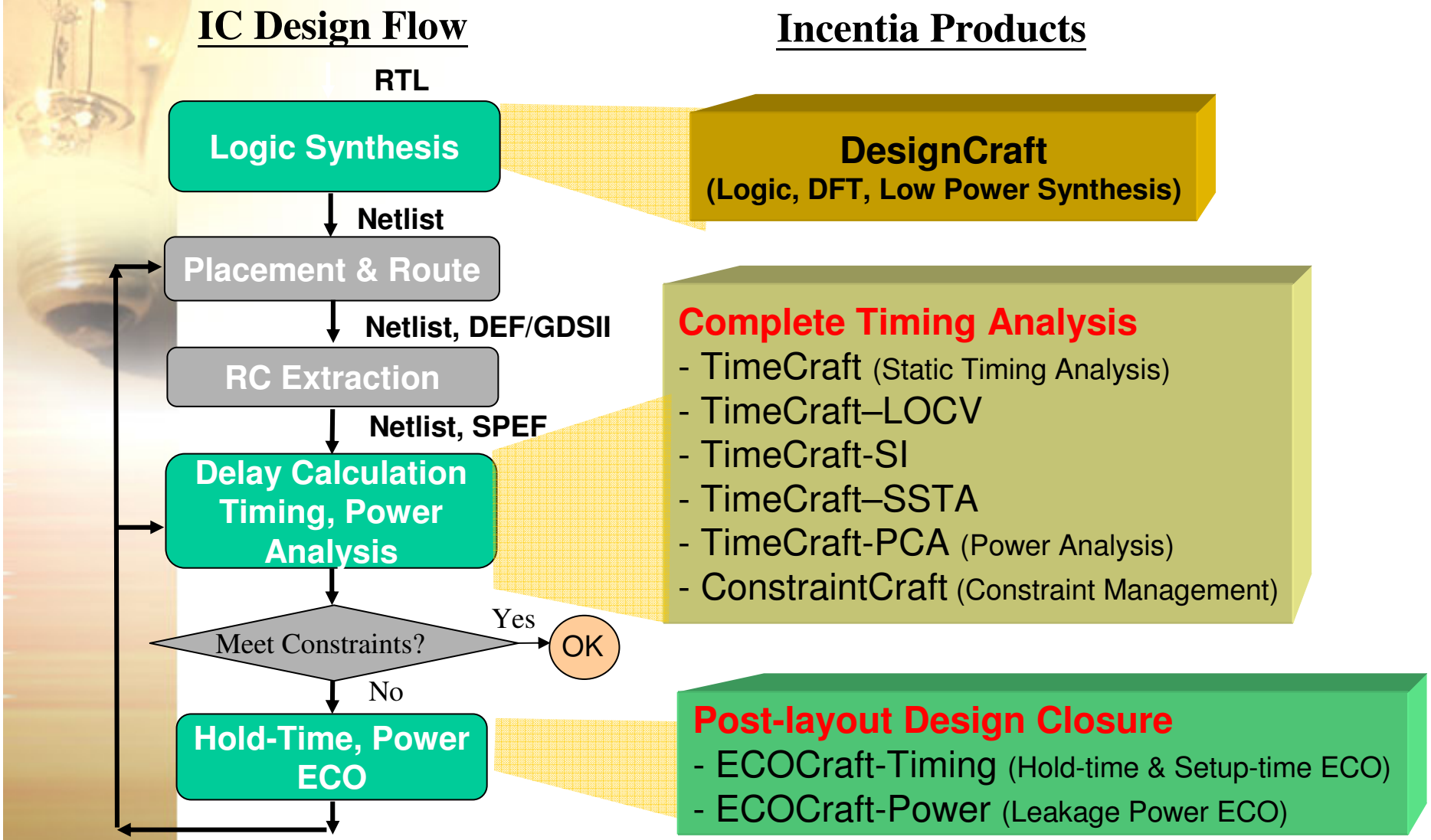
Direct Offices:
USA: Silicon Valley
Southern California
Taiwan: Hsinchu

Distributors:
Japan (Marubeni), India (ICON),
China (OnePass Solutions),
Korea (ED&C), Israel (AST)

Incentia Product Offering

- **Logic, Low Power, DFT Synthesis solution**
  - **DesignCraft, PowerCraft, TestCraft**
- **Timing analysis solution**
  - TimeCraft: World fastest Static Timing Analyzer (STA)
  - TimeCraft-LOCV: Location Based OCV
  - TimeCraft-SSTA: Statistical STA
  - TimeCraft-SI: Signal Integrity
  - TimeCrfaft-PCA: Power Analysis
  - ConstraintCraft: Constraint Management
- **Design closure solution**
  - ECOCraft-Timing: Hold-time & Setup-time ECO
  - ECOCraft-Power: Leakage power ECO

How Incentia Products Fit into Design Flow

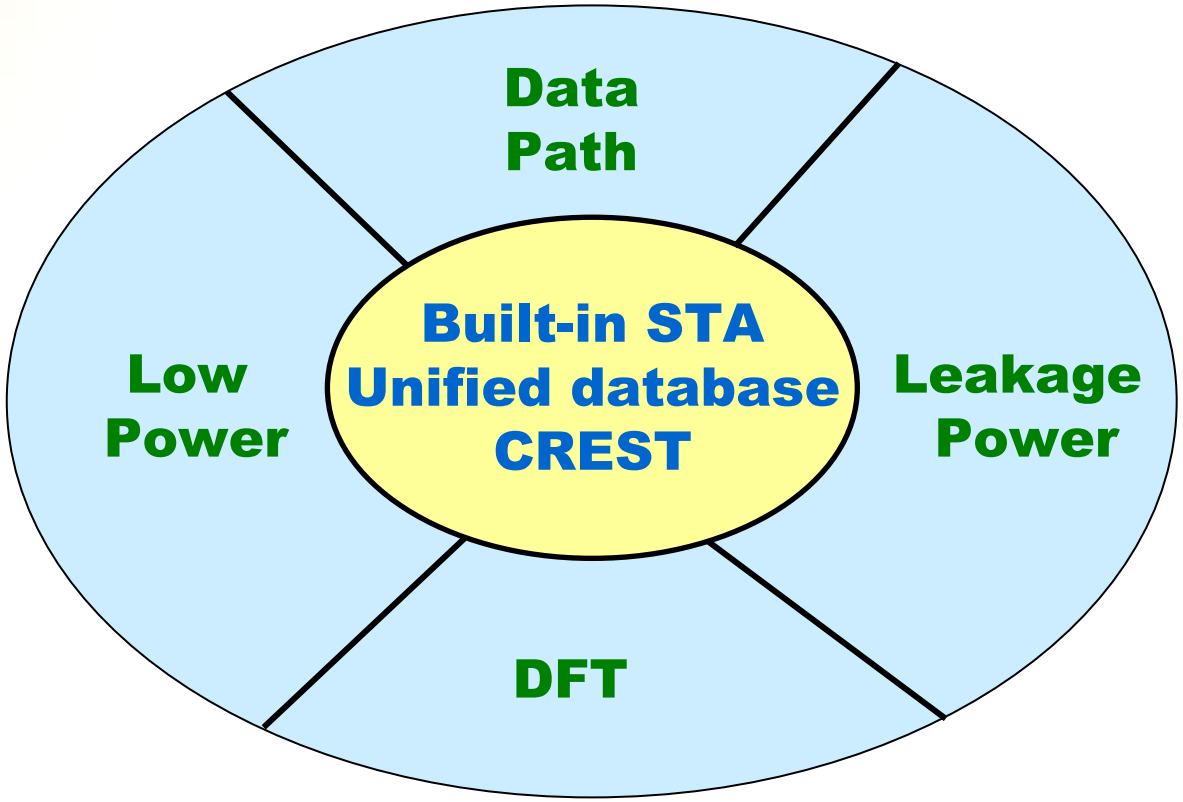


DesignCraft Highlights

- Complete solution
  - Logic, low power, DFT
  - Fast synthesis runtime: Up to 3X faster than other solutions!
  - Specialized for aggressive area & power reduction
- Support all the standard inputs
  - Verilog/VHDL
  - Synopsys .lib, CCS library
  - Timing constraints in SDC
  - Tcl scripts
- Easy to migrate and use!
  - Take your RTL code directly without any modifications!
  - Very similar script usage
- **Much more affordable solution with better results!**
- **Proven through many customer tape-outs**

Complete Integrated Logic Synthesis

DesignCraft



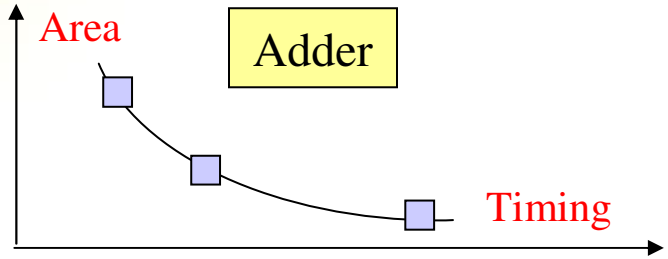
Target for designs demanding aggressive area & power reduction

Contents

- Data-path synthesis
- Optimization strategy
- Low power synthesis
  - Multi-voltage designs
  - Dynamic power optimization
  - Leakage power optimization
- Power analysis
- DFT synthesis

Data-path Synthesis

- Data-path (e.g. adder, multiplier, …) implementation architectures affect synthesis area / timing QOR
  - Various architectures are available for area/timing tradeoff



Implementation	Description
rpl	Ripple carry
bk	Brent-Kung carry look-ahead
fbk	Fast Brent-Kung carry look-ahead

- Data-path inference in RTL code
  - Automatic inference of data-path & architecture
  - Explicit instantiation through Incentia Parameterized Components
- Data-path resource sharing
  - Control through compiler derivatives

Incentia Parameterized Components (IPC)

- Support more than 40 pre-defined data-paths through IPC
  - add, sub, addsub, mult, divider, inc, dec
  - absval, ash, bsh, shifter, decoder
  - mac, prod_sum, square, sqrt, …
  - Also pipe-lined IPCs
- Data-path instantiations in RTL code
  - Using IPC notations
  - Can also take your existing synthesizable RTL code directly without any modifications!

Data-path Synthesis: Inference Example

- Automatic inference from RTL code
  - Automatic architecture selection and optimization

```
module top (a, b, ci, sum, co);
input [15:0] a, b;
input ci:
output {15:0] sum;
output co;
assign {co, sum} = a + b + ci;
endmodule
```

- Explicit instantiation
  - Instantiate pre-defined data-paths (adder, multiplier, etc)
  - Optionally specify the implementation architecture through Verilog/VHDL compiler directives
  - Also take DW*** from your existing RTL code

```
module top (a, b, ci, sum, co);
parameter width = 16;
input [width-1:0] a, b;
input ci:
output [width-1:0] sum;
output co;
// incentia ipc_implementation fbk I1
ipc_add #(width) I1 (.A(a), .B(b), .CI(ci),
.SUM(sum), .CO(co));
endmodule
```

Implementation	Description
rpl	Ripple carry architecture
bk	Brent-Kung carry look-ahead architecture
fbk	Fast Brent-Kung carry look-ahead architecture

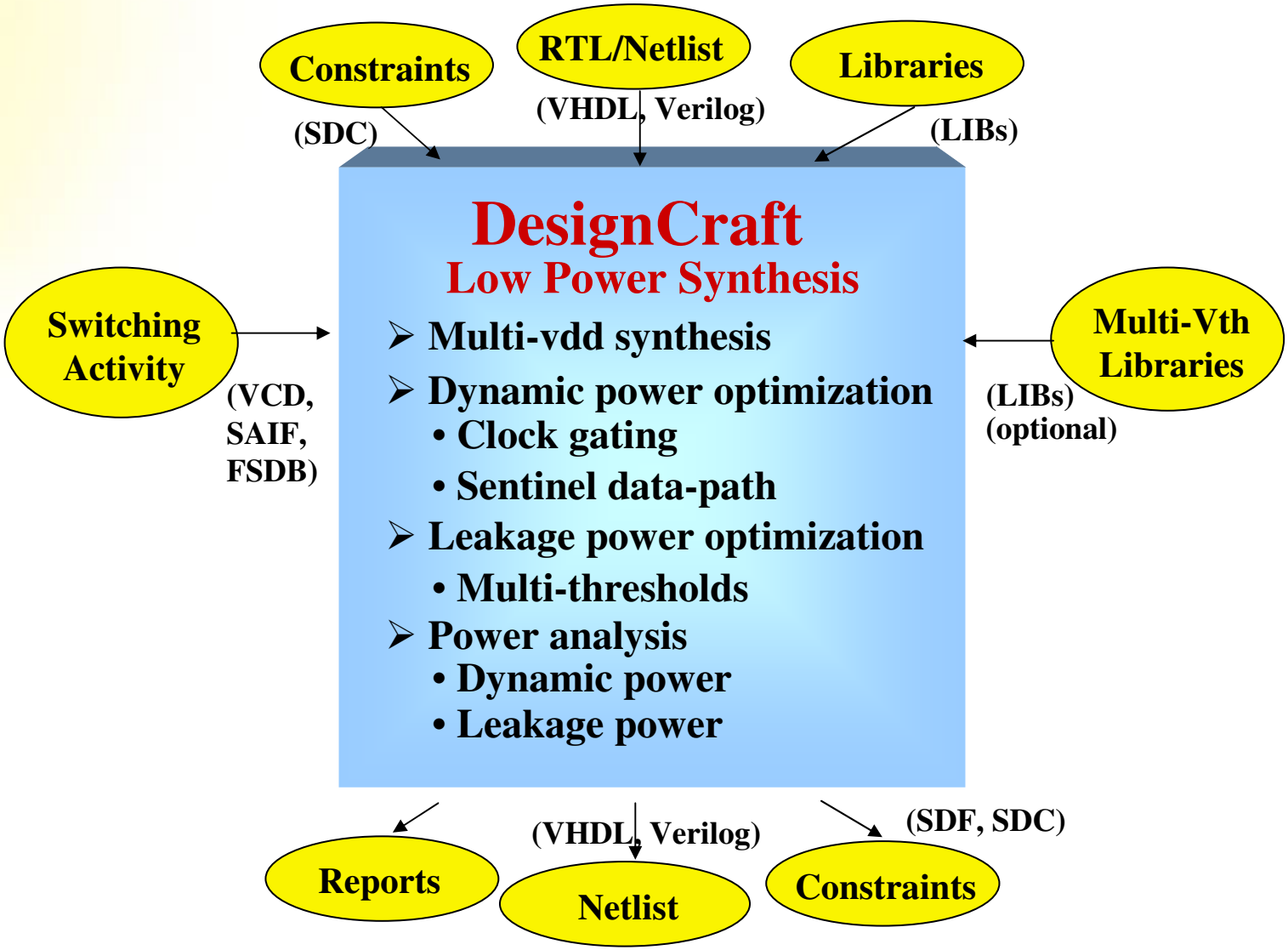
Optimization Strategy

- Synthesis and optimization are done based upon constraints
  - Finding the best trade-off point for timing, area, and power

```
optimize ?-mode mode_type? ?-incremental?
         ?{ -only_design_rule | -no_design_rule }?
         ?-only_fix_hold? ?-expand expand_type?
         ?-structure_effort structure_effort_type?
         ?-map_effort map_effort_type?
         ?-demote_timing? ?-scan? ?-boundary_opt?
         ?-leakage_power?
```

- Options to explore for best timing, area or power, and runtime
  - mode: timing vs. area optimization mode
  - structure_effort: boolean optimization for area mode
  - map_effort:
  - demote_timing: another option for
  - boundary_opt: allows for boundary optimization for better quality

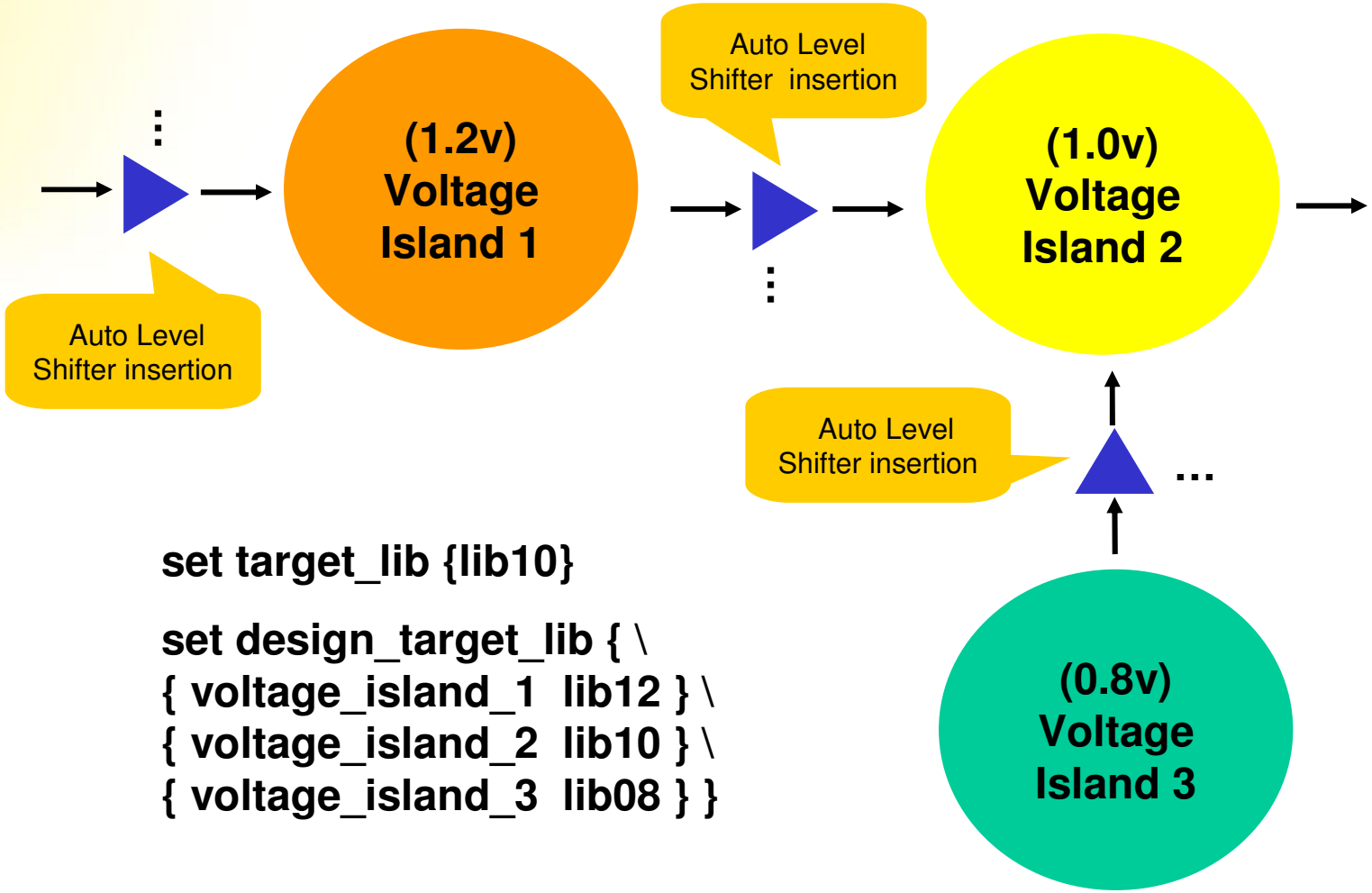
Low Power Solution



Multi-vdd Synthesis

- Support of both top-down and bottom-up approaches
- Automatic level shifter insertion among different voltage domains
- Seamless integration with DFT
- Concurrent multi-vdd and multi-threshold optimization
- CPF support
- Support multi-bit sequential cells for area/power reduction
- Efficient low power cell (level-shifter, isolation/retention cell) insertion in early or integration stage

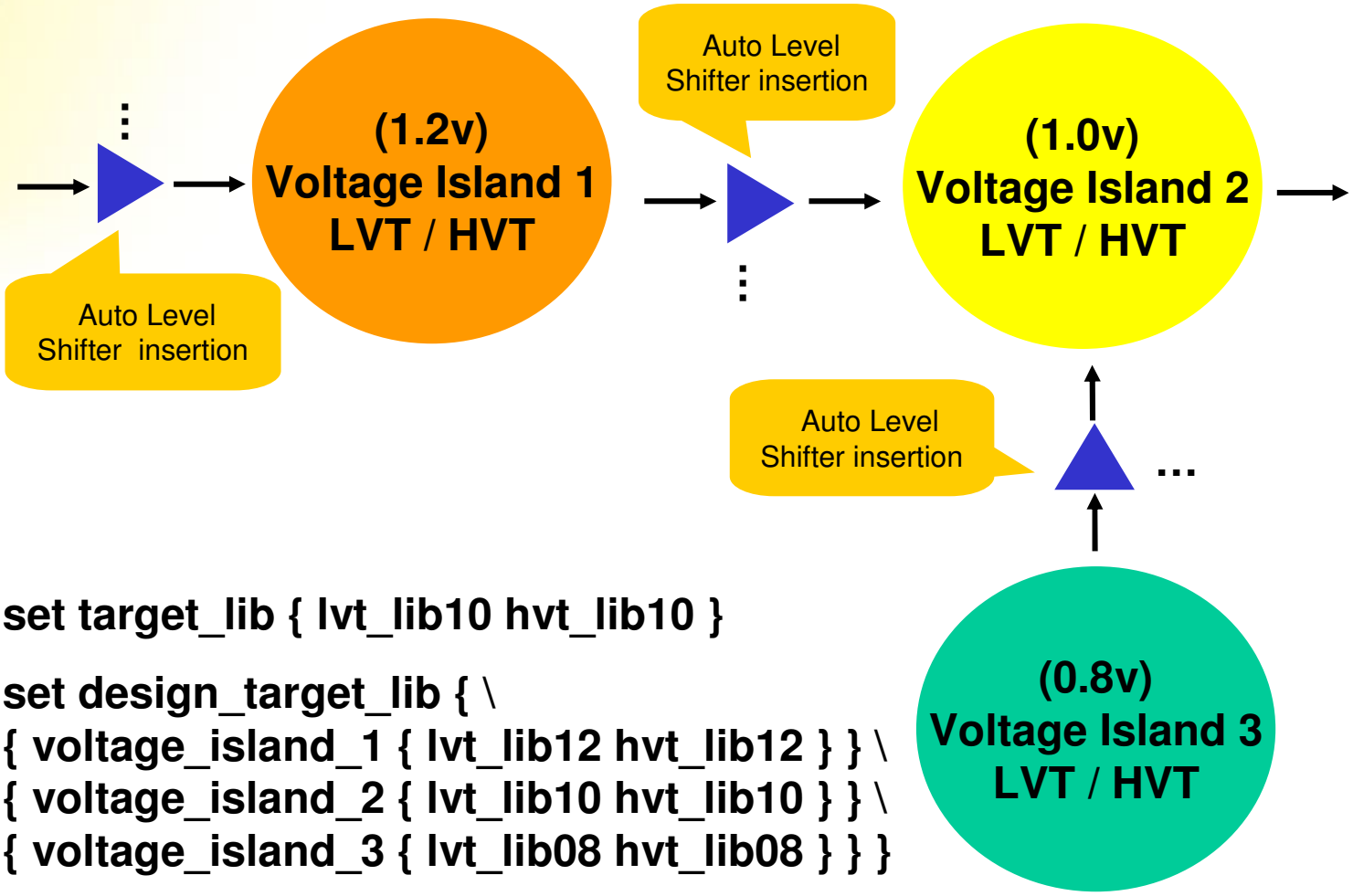
Automatic Level Shifter Insertion




```
set target_lib {lib10}

set design_target_lib { \
{ voltage_island_1  lib12 } \
{ voltage_island_2  lib10 } \
{ voltage_island_3  lib08 } }
```

Concurrent Multi-vdd Multi-threshold



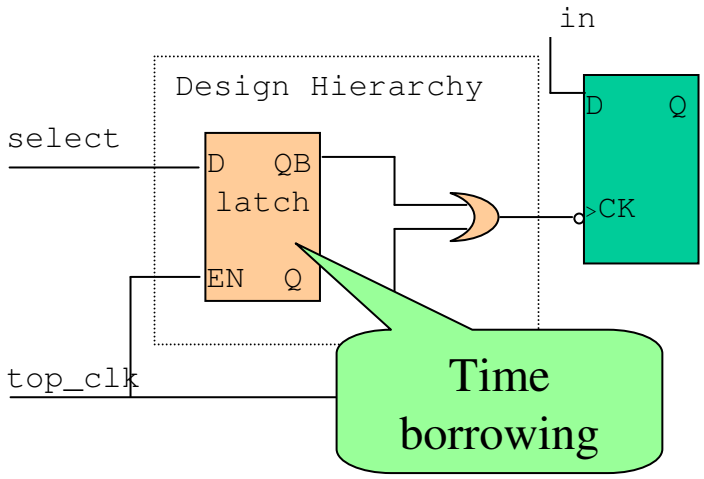

```
set target_lib { lvt_lib10 hvt_lib10 }

set design_target_lib { \
{ voltage_island_1 { lvt_lib12 hvt_lib12 } } \
{ voltage_island_2 { lvt_lib10 hvt_lib10 } } \
{ voltage_island_3 { lvt_lib08 hvt_lib08 } } }
```

Dynamic Power Optimization

- Clock gating
  - Do not depend on specific RTL coding styles: highest achievable rate for clock gating implementation

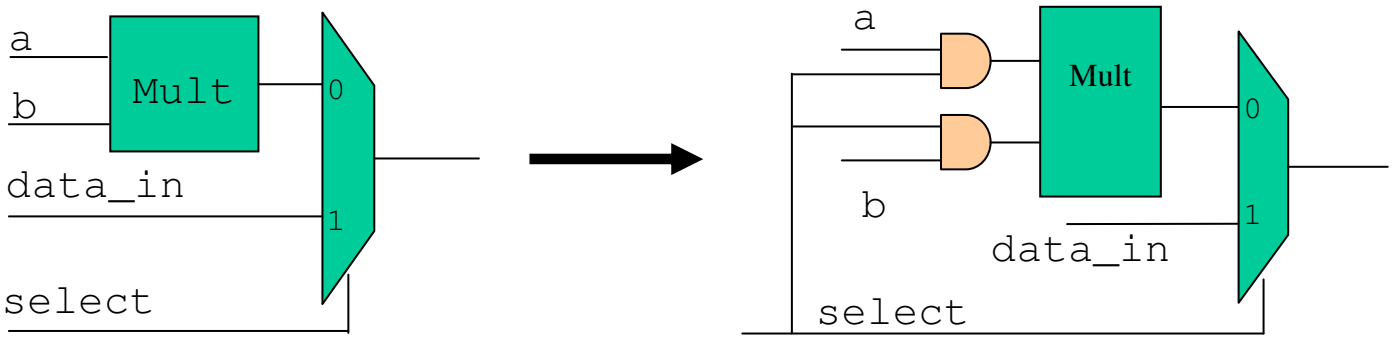
```
always @ (negedge clk)
  if (select)
    out <= in;
```



  - Unique latch-based clock gating optimization: take full advantage of time borrowing to meet timing and reduce area
  - Able to use integrated clock gating cells in the library

Dynamic Power Optimization (Cont)

- Sentinel data path for sleep mode operation
  - Generate guarding sentinel logic for data-paths
  - RTL compiler directive controls



Dynamic Power Optimization: Customer Cases

- Clock gating can reduce about 30% dynamic power

Designs	Power Consumption (mW)		Power Savings
	No Clock Gating	Clock Gating	
Design 1	330	220	33.3%
Design 2	638	447	29.8%
Design 3	736	489	33.5%
Design 4	1120	765	31.7%

Leakage Power Optimization

- Global leakage power optimization
  - Critical paths: low-vt cells
  - Non-critical paths: high-vt cells
- Flexible controls of optimization priority
  - Timing first: reduce leakage power while meeting timing constraints
  - Leakage power first: reduce leakage power as much as possible while keeping timing results in reasonable range
- End results
  - An average of 30% leakage power reduction
  - Up to 80% leakage power reduction using aggressive leakage power optimization

Leakage Power Optimization

	Leakage Power Optimization Effort	Leakage Power Reduction	Timing (WNS)
Design1 (0.13nm, 2M gates)	None		0.08
	Normal	30.8%	0.00
	Aggressive	68%	-0.02
Design2 (90nm, 4M gates)	None		0.02
	Normal	32.3%	0.00
	Aggressive	76.4%	-0.08

Power Analysis Highlights

- Inputs
  - Netlist or RTL
  - Activity file formats: VCD, SAIF, FSDB
  - Timing data: SPEF, annotated cap, wire-load model
- Reports
  - Dynamic power: Switching, Internal (SDPD), Glitch, X-transition power
  - Static/Leakage power (SDPD)
  - Average, peak, clock tree, interval power
- Peak power output for waveform viewer
- Very fast runtime: 2X to 5X faster than other solutions!

Power Analysis: Switching Activities

- Vectorless power analysis capability
  - No need of input switching activity
- Gate-level power analysis with RTL-input activity file
  - Skip long gate-level simulation for large gate-level activity file
- Analyze sub-design power with top-level design activity files

```
compile_fsdb –current_design current_subckt –strip_path xxxx
```

- Multiple activity files for each sub-design

```
compile_fsdb –multi_acitivity_file  { { top/I1 I1.fsdb U1} { top/I1 out2.dump U2} }
```

- Multiple activity files for different time slots

```
compile_fsdb –strip_path top/I1 "file_1.fsdb file_2.fsdb file_3.fsdb“
```

Power Analysis: Reports

- Dynamic, static (leakage)
- Average, peak, clock tree
- Interval power
  - Report average power for interested intervals

Single interval:

```
set_pa_option –interested_interval “100 200”
compile_fsdb or compile_vcd
report_analyzed_power  –interested_interval “100 200” > power_1.rpt
```

Multiple intervals:

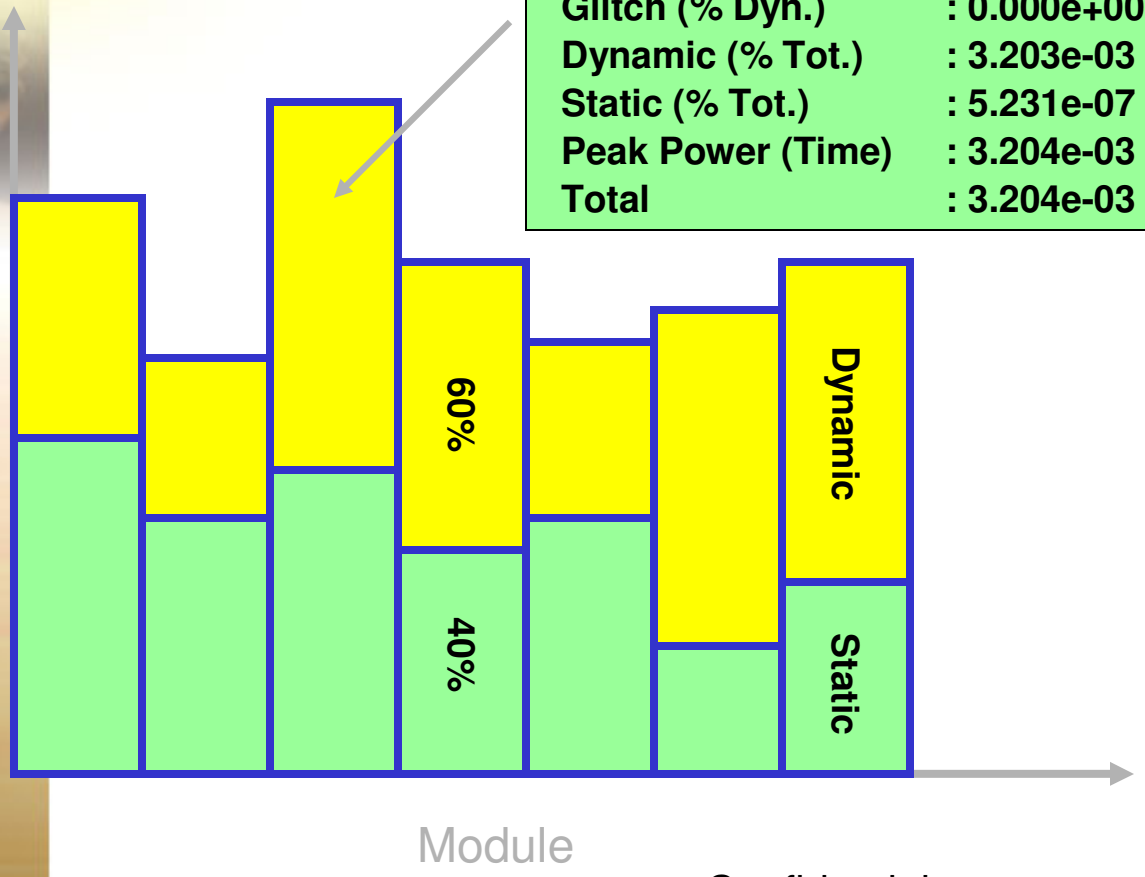
```
set_pa_option –interested_interval “100 200 300 400 500 600 700 800”
compile_fsdb or compile_vcd
report_analyzed_power  –interested_interval “100 200” > power_1.rpt
report_analyzed_power  –interested_interval “300 400” > power_2.rpt
report_analyzed_power  –interested_interval “500 600” > power_3.rpt
report_analyzed_power  –interested_interval “700 800” > power_4.rpt
```

- Toggle rate report

Power Distribution Diagram

Module: sub1

Switching (% Dyn.)	: 1.624e-03 (50.70%)
Internal (% Dyn.)	: 1.579e-03 (49.30%)
X-trans (% Dyn.)	: 0.000e+00 (0.00%)
Glitch (% Dyn.)	: 0.000e+00 (0.00%)
Dynamic (% Tot.)	: 3.203e-03 (99.98%)
Static (% Tot.)	: 5.231e-07 (0.02%)
Peak Power (Time)	: 3.204e-03 (20345 ~ 20355)
Total	: 3.204e-03



Power Type

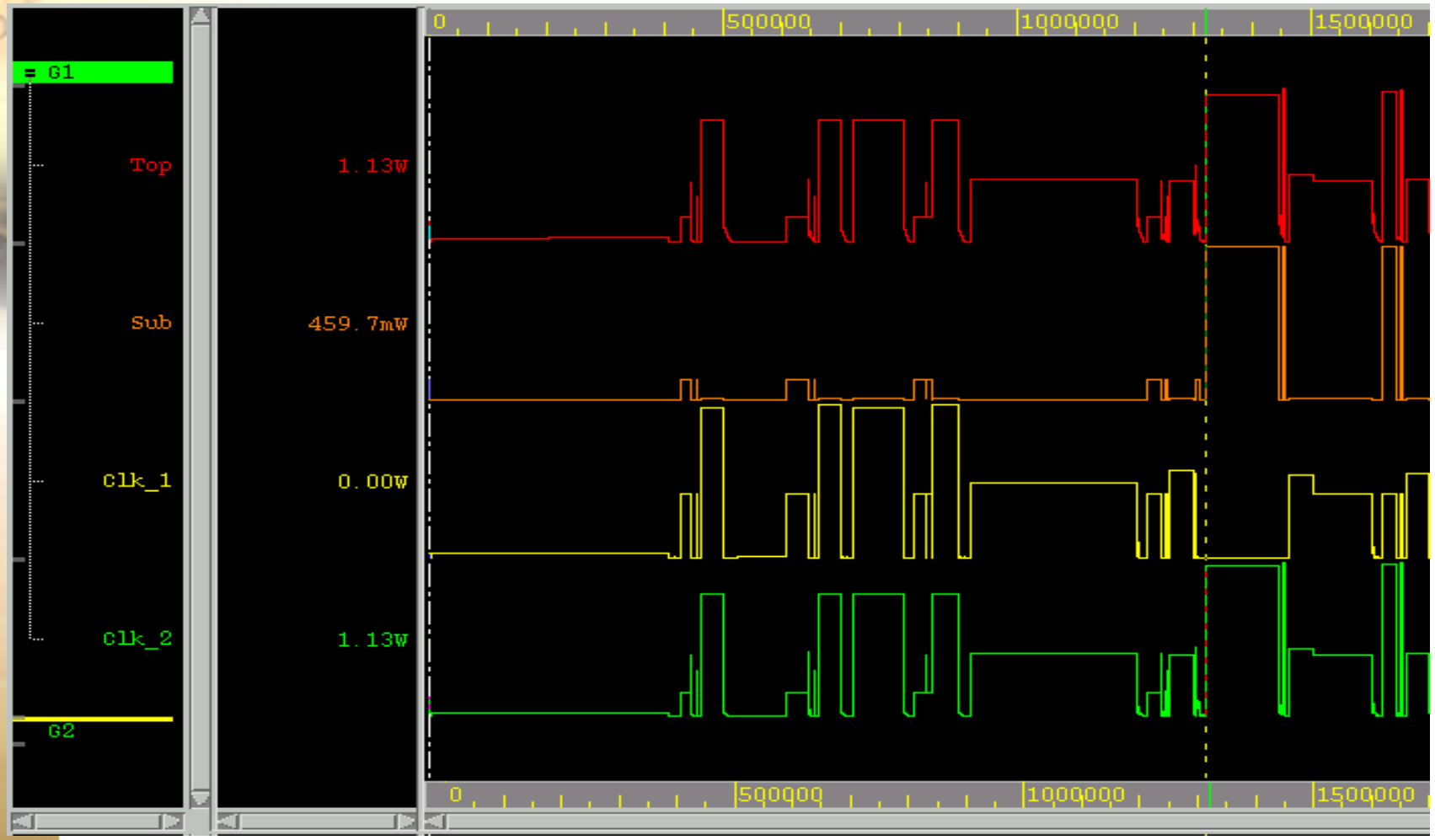
- Dynamic
- Static

Power Scope

- Clock
- Data

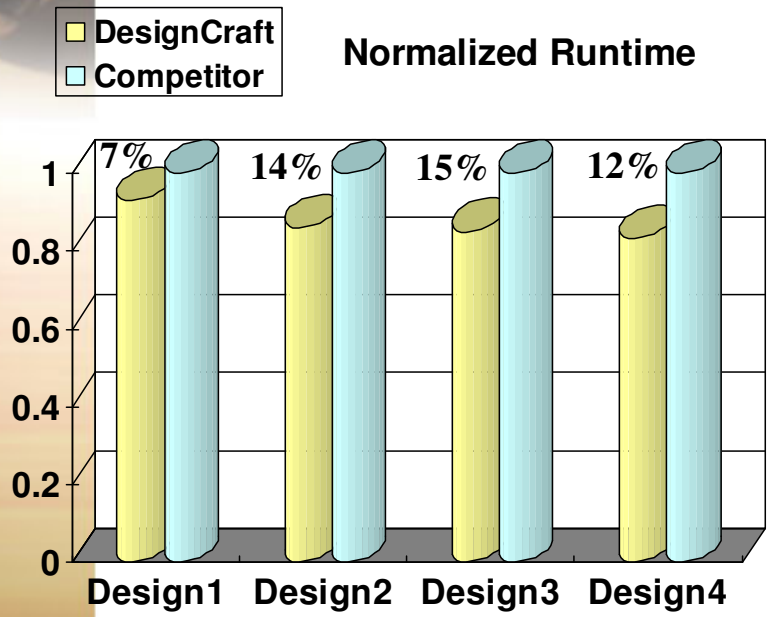
Time-Base Power Waveform in nWave

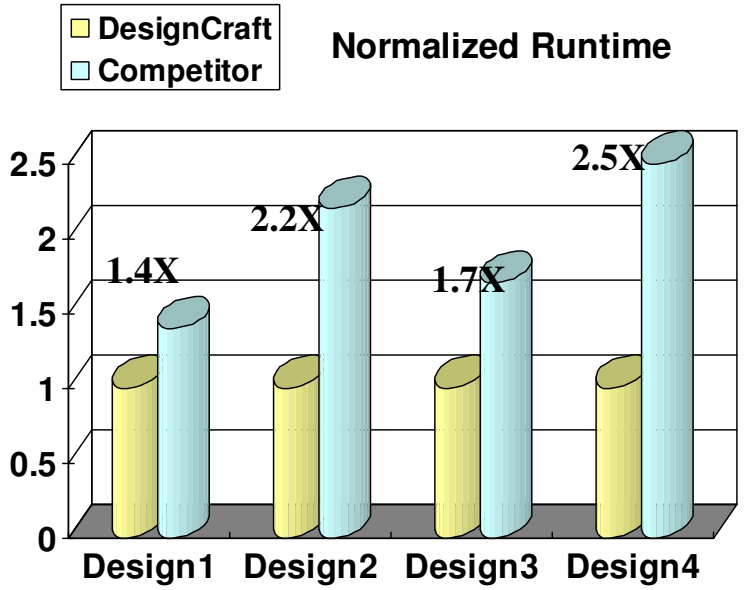
- Output peak power info in VCD or FSDB format



DesignCraft Benchmark Data

Design1	1.2M logic gates, 8 macros; 133 MHz; .130nm
Design2	2.2M logic gates; 12 macros; 250 MHz; 90nm
Design3	3.6M logic gates; 24 macros; 250 MHz; 90nm
Design4	5.6M logic gates; 22 macros; 350 MHz; 65nm





Summary

- Complete integrated synthesis solution
  - Logic, Low Power
- Very fast runtime with big capacity
  - Up to 3X faster than other solutions!
- Most aggressive reduction in chip area and power consumption
  - Smaller synthesized gate counts: up to 20% less
  - Smaller power: up to 20% less
- Many customer tape-outs in different applications
  - Communication, mixed-signal, networking, wireless, consumer electronics, multi-media, graphics
  - From 250nm to 65nm
- Best performance-price synthesis solution in market!