SYNOPSYS®
Predictable Success

IC Compiler 2010.03 Incremental Training

Design Planning: Flows

Agenda

- Flows
  - Hierarchical Reference Methodology updates
  - Channeled, narrow channel, and abutted hierarchical flows
  - Exploration flow
  - On Demand Loading flow

Major Updates in 2010.03 IC Compiler Design Planning Flows

- Support flip-chip flow for flat floorplans
- Configurations for hierarchical floorplan styles:
  - Channeled
  - Narrow channeled
  - Abutted
- Configurations for hierarchical design styles:
  - Multiply instantiated modules (MIM)
  - Black box

Table of Supported Hierarchical Flow Combinations in Reference Methodology

	Regular Design Styles	Black Box	MIM
Channeled	Yes	Yes	Yes
Narrow Channeled	Yes	No	No
Abutted	Yes	No	No

New Options on SolvNet RMgen for 2010.03 IC Compiler Reference Methodology

- Hierarchical Design Planning options:
  Floorplan Style: DEFAULT(i.e. channeled) / NARROW_CHANNELED / ABUTTED
  Design Style: NONE(i.e. regular) / BLACK_BOX / MIM
  (Design Style option is only available if you select DEFAULT for Floorplan Style option

Hierarchical Design Planning

Flow Changes

- Overview
  - In the 2010.03 hierarchical design planning flow, timing optimization (in-place optimization) is moved after the pin assignment step.
    The new flow can efficiently optimize timing violations on feedthrough paths and achieve better timing QoR
  - Hierarchical flow uses Zroute for congestion analysis and pin assignment

- UI
  - Refer to hierarchical flow chart

2010.03 Hierarchical Design Planning Flow

- Read netlist
- Initialize floorplan
- Virtual flat placement
- Plan groups and shaping
- Placement
- PG connections and PNS
- Prototype routing
- High-fanout synthesis
- Mark clock tree
- Plan-group-aware routing
- Analyze and finalize Routing
- In-place optimization
- Timing budgeting on plan groups
- Commit hierarchy

```
read_verilog

initialize_floorplan

create_fp_placement

shape_fp_blocks

create_fp_placement
derive_pg_connect / synthesize_fp_rail/
commit_fp_rail

route_zrt_global -effrot minimum

optimize_fp_timing -hfs_only

mark_clock_tree -clock_net
set_route_zrt_common_options -plan_group_aware
all_routing
route_zrt_global -effort low
analyze_fp_routing -finalize {plan_groups}

optimize_fp_timing

allocate_fp_budgets

commit_fp_plan_groups
```

Command Changes in Flow (2010.03)

Channel / Near Abutted / Abutted

Floorplan type	High-fanout synthesis	In-place optimization
Channelled	`optimize_fp_timing -hfs_only`	`optimize_fp_timing`
Narrow Channel	`optimize_fp_timing -hfs_only`	`optimize_fp_timing`
Abutted	`optimize_fp_timing -hfs_only -no_new_cells_at_top_level`	`optimize_fp_timing -no_new_cells_at_top_level`

2010.03 Hierarchical Design Planning Flow

- User Benefit
  - 2010.03 flow provides better timing QoR and faster flow runtime
- Flow Recommendation
  - 2010.03 reference methodology script uses the new hierarchical flow
  - Flow changes apply to all hierarchical design flows (channel, narrow channel, abutment, MIM)

Exploration Flow

- Overview
  - Provides a way to quickly develop a floorplan and analyze the routability
- UI
  - Placement:
    - `create_fp_placement -exploration`
  - Routing:
    - `set_route_zrt_common_options -plan_group_aware_quality exploration`
    - `route_zrt_global`
  - In-place optimization:
    - `virtual_ipo`
    - `virtual_ipo -end`
  - Timing budgeting:
    - `allocate_fp_budgets -exploration`

Placement Options and Strategies That Cannot Be Used in the Exploration Flow

- Options
  - `-effort`
  - `-incr`
  - `-cong`
  - `-timing`
  - `-optimize_pins`
- Placement Strategies
  - `-force_auto_detect_hierarchy`
  - `-congestion_effort`
  - `-adjust_shapes`
  - `-legalizer_effort`
  - `-spread_spare_cells`
  - `-virtual_IPO`
  - `-pin_routing_aware`

Exploration Flow

- User Benefit
  - 2x faster turnaround time (TAT) as compared to the regular design planning flow
- Flow Recommendation
  - Flow diagram on the next slide

Regular Design Planning Flow:

Version 2010.03

Timing-Driven Regular Design Planning Flow

Floorplan analysis

Read netlist	`read_verilog`
Initialize floorplan	`initialize_floorplan`
Virtual flat placement (MoE=0)	`create_fp_placement`
Plan groups and shaping	`shape_fp_blocks`
Placement (MoE = Auto / On)	`create_fp_placement`
Derive PG connect ions/ PNS	`derive_pg_connect / synthesize_fp_rail / commit_fp_rail`
High-fanout synthesis	`optimize_fp_timing -hfs_only`
Mark clock tree	`mark_clock_tree`
Plan-group-aware routing	`set_route_zrt_common_options - plan_group_aware_quality regular - plan_group_aware < all_routing | top_level>` `route_zrt_global` *(routing will be done on a legalized cells)*
Analyze and finalize routing	`analyze_routing -finalize`

In-place optimization (trace or regular mode)	`set_fp_tracemode (optional)`
	`optimize_fp_timing` *(database and netlist update will be performed)*
	`end_fp_trace_mode (optional)`

Timing budgeting on plan groups	`allocate_fp_budgets`
Commit plan groups	`commit_fp_plan_groups`

Exploration Flow in Design Planning

Timing-Driven Exploration Design Planning Flow

Floorplan analysis

Read netlist

```
read_verilog
```

Initialize floorplan

```
initialize_floorplan
```

Virtual flat placement (MoE=0)

```
create_fp_placement -exploration
    (clumped, non-legalized placement with HM's overlapped)
shape_fp_blocks
```

Plan groups and shaping

```
set_fp_placement_strategy -name hfn_are_interface -value 1
create_fp_placement -exploration
    (clumped, non-legalized placement with NO HM's
    overlapped)
```

Placement (MoE = Auto / On)

Derive PG connections / PNS

```
derive_pg_connect / synthesize_fp_rail / commit_fp_rail
```

High-fanout synthesis

```
set_ahfs_options -optimize_buffer_trees true -incremental
    true
set fpopt_no_legalize true
optimize_fp_timing -hfs_only
```

Mark clock tree

```
mark_clock_tree
```

Plan-group-aware routing

```
set_route_zrt_common_options -plan_group_aware_quality
    exploration  -plan_group_aware < all_routing |
    top_level>
route_zrt_global (routing will be done on a non-legalized
    cells)
```

Analyze and finalize routing

```
analyze_routing -finalize
```

Feedthrough in-place optimization

```
Optimize_fp_timing - feedthrough_buffering_only
```

Virtual IPO (trace or regular mode)

```
set_fp_tracemode (optional)
virtual_ipo (database / netlist update will not be
    performed)
virtual_ipo -end
end_fp_trace_mode (optional)
```

Timing budgeting on plan groups

```
allocate_fp_budgets -exploration
```

Commit plan groups

```
commit_fp_plan_groups
```

Exploration Flow

- Limitations
  - The following design styles are not supported in this release
    - Multicorner-multimode
    - Multiply instantiated modules
    - Black box

Exploration Flow

Summary

- Overall turn around time is 2X faster than regular design planning flow
- Works for channeled or abutted hierarchical design styles
- Very useful during initial prototype iterations to explore a large solution space in a given amount of time

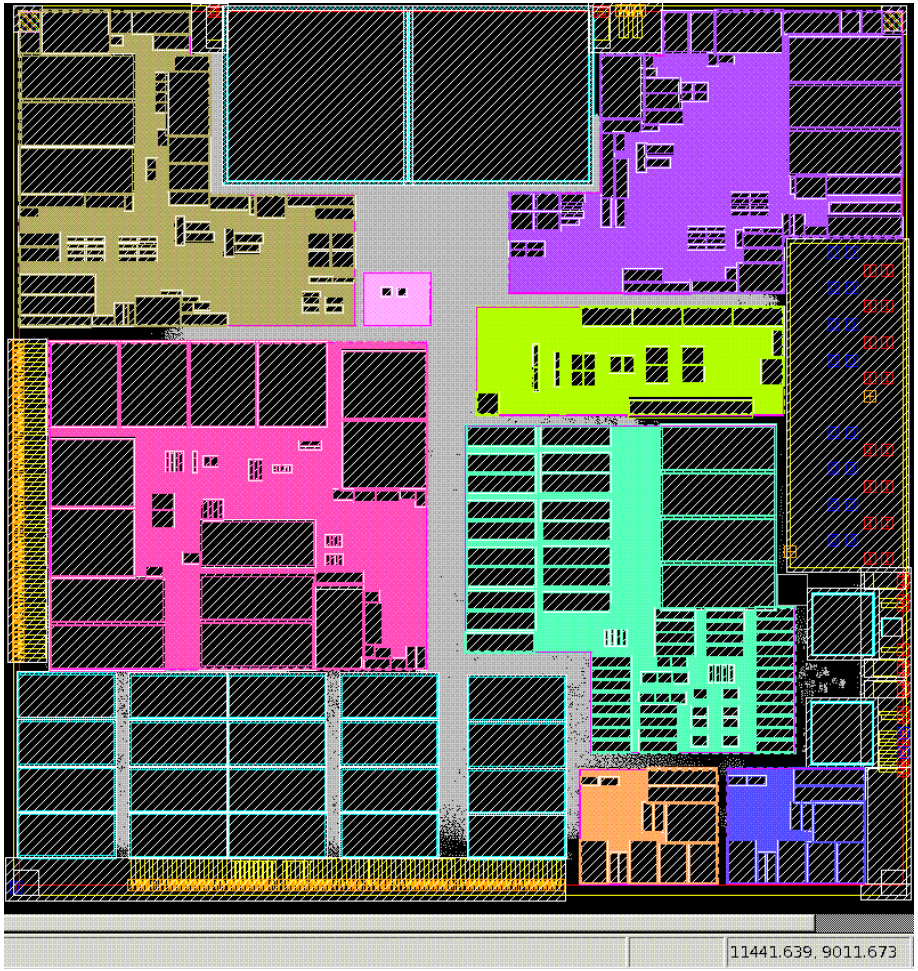
On Demand Loading (ODL) Overview

- On Demand Loading enables a designer to interactively run multiple top-level design planning experiments in a single work day
  - Abstracts the design so a much smaller memory footprint is required for top-level design planning
  - Plan group contents are reduced to interface logic, however all macro cell instances are retained
  - All top-level logic is also maintained
- Intelligent abstraction enables users to assess and modify plan group shapes and interface timing so that users can converge on the final top-level floorplan much faster
- Important outputs of ODL are:
  - Block budgets & boundaries
  - Pin assignment & feedthroughs
  - Placement information of top level blocks

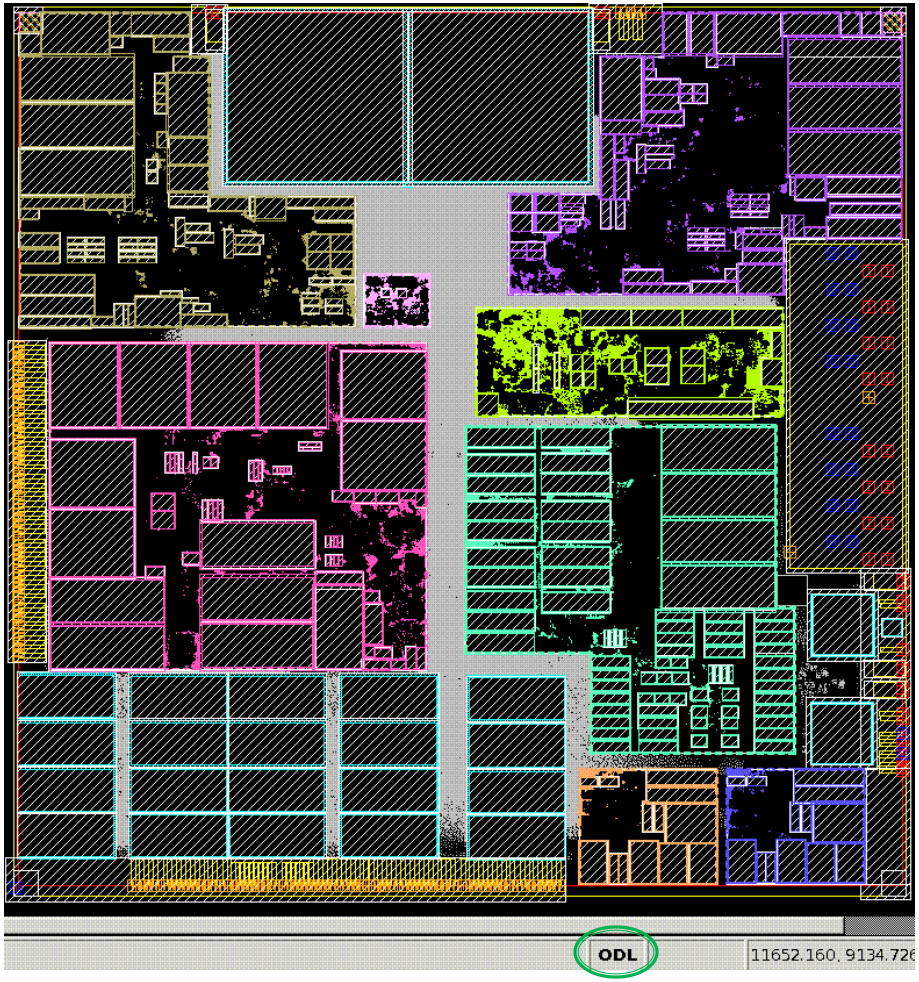
On Demand Loading

Plan Group Abstraction (1.815M -> 0.048M cells)

Full Design



On Demand Netlist



On-Demand-Loading Flow

- UI
  - New commands

    ```
    create_on_demand_netlist
    report_on_demand_netlist
    remove_on_demand_netlist_data
    query_on_demand_netlist
    ```

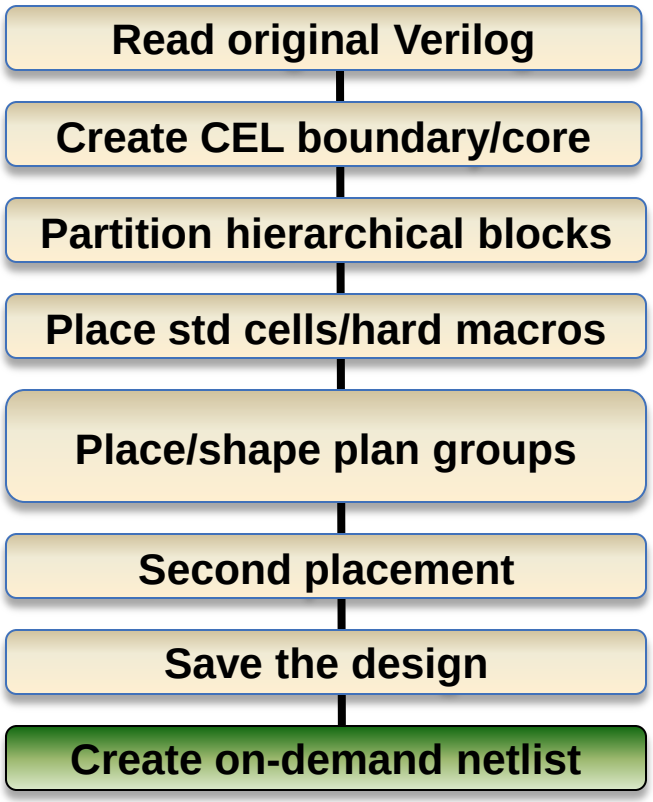
  - Changed commands

    ```
    allocate_fp_budgets
    commit_fp_plan_groups
    ```

On-Demand-Loading Flow

- User Benefit
  - Since design planning is iterative in nature, reducing the amount of design data can significantly speed up the design planning process
    - 2X - 10X speed up in turnaround time can be achieved
    - 2X - 5X memory reduction is possible
  - QoR neutral (not improved or degraded)
- Flow Recommendation
  - If you have a Milkyway design that has gone through partitioning, placement, and shaping (you have a virtual flat design that has plan groups inside the core area and has completed placement, including hard macros), you can use the on-demand-loading flow

Create Initial Floorplan and On-Demand Netlist

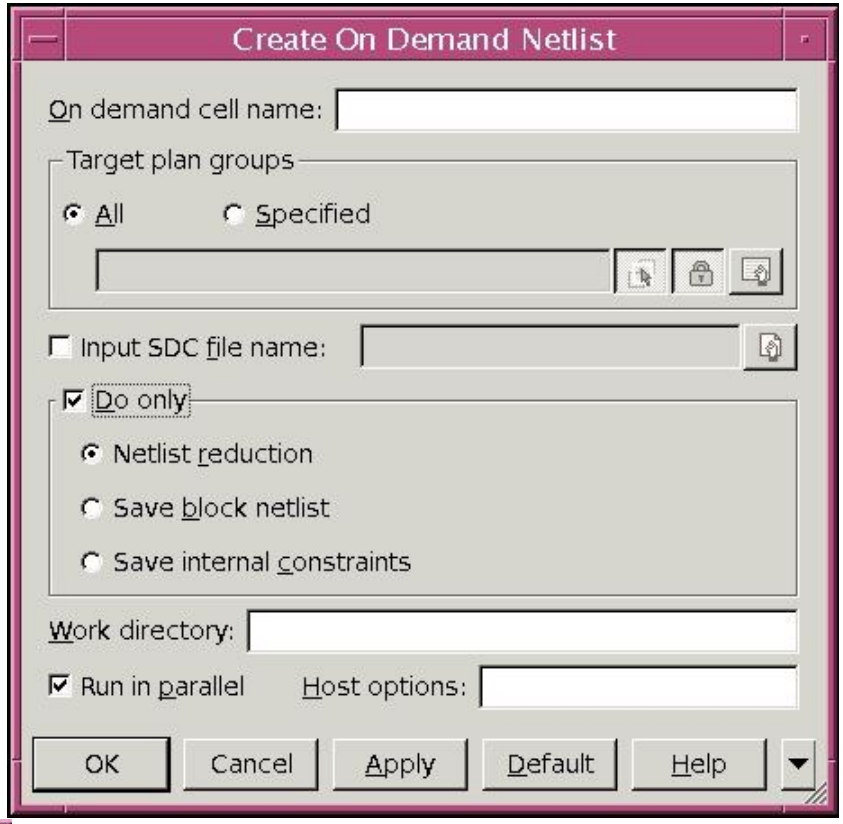


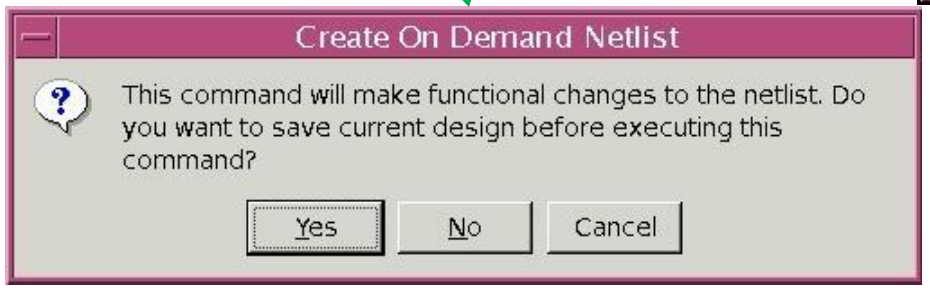
- Use the full netlist to estimate the plan group sizes
- Use non-timing-driven functions for initial floorplanning
- Create on-demand netlist
  - Abstracts plan group netlists
  - Saves internal plan group timing constraints

```
create_on_demand_netlist
```

Using the GUI to Create an On-Demand Netlist

If you create an on-demand netlist using the GUI (Partition > Create On Demand Netlist), you must go through an extra dialog box





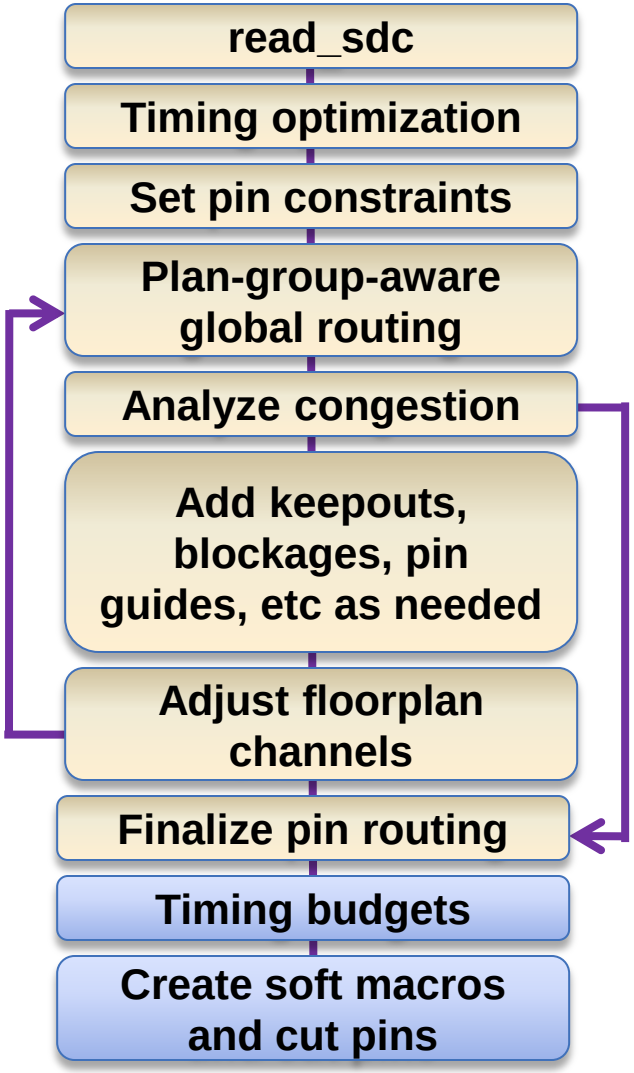
GUI Indications That a Design Is Represented by an On-Demand Netlist

Infotip of the plan group indicates that it is an on-demand netlist

Tells you the current design uses on-demand loading (it is an on-demand netlist)

On-Demand-Loading Flow

Using the On-Demand Netlist



- Run the virtual flat top-level design planning flow with the on-demand netlist
- Optimize plan group shapes and channel widths during design planning
- As the number of iterations increase, the TTR for the ODL flow significantly decreases compared to the standard VF flow

```
allocate_fp_budgets
commit_fp_plan_groups
```

On-Demand-Loading Flow

Enhancements and Limitations

- Future on-demand-loading flow enhancements
  - MultiCorner-MultiMode support
  - Multiply Instantiated Module usage
  - PNA/PNS support
  - Using `derive_pg_connection` on an on-demand netlist
  - UPF support
- Known Limitations
  - Hierarchical manipulation of an on-demand netlist is not supported
  - Clock tree planning is not supported with an on-demand netlist
  - DFT operations are not supported with an on-demand netlist
  - You cannot use most non-design-planning commands

Summary of Design Planning Updates

Module 4

- Hierarchical reference methodology changes that result in better timing QoR and faster flow runtime in 2010.03 flow
  - Flow changes apply to all hierarchical design flows (channeled, narrow channeled, abutted, MIM)
- Fast Time-to-results (TTR) Improvements
  - Exploration flow (for initial prototyping)
  - On-Demand-Loading flow (once you have an initial floorplan)

Predictable Success