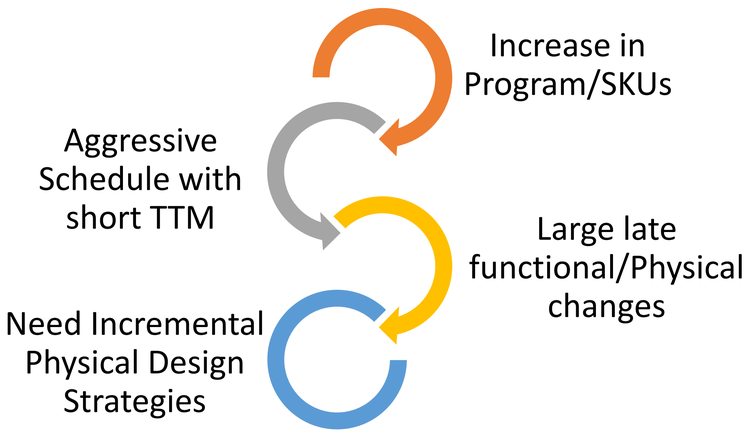
The Anatomy of Incremental Floorplanning Solutions

Sivaramakrishnan Harihara Subramanian, Venkatesh RS
Intel Corporation, Folsom, CA

Overview

Problem Statement

Increased GPU Product segments (Gaming/ML/Video)



Incremental Floorplanning Solutions

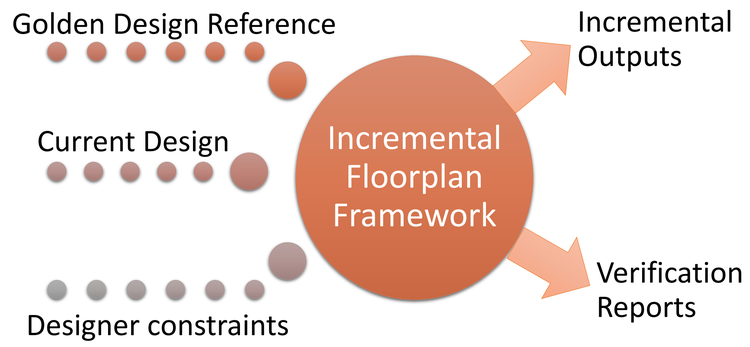
- Retention strategies to maximize reuse of recipes
- Avoid block resets through streamlined 'parent' incremental floorplanning techniques
  - ❖ ↑ Schedule efficiency - weeks per block ↓
  - ❖ ↑ Resource efficiency – gates per designer ↑

Goals and I/O map

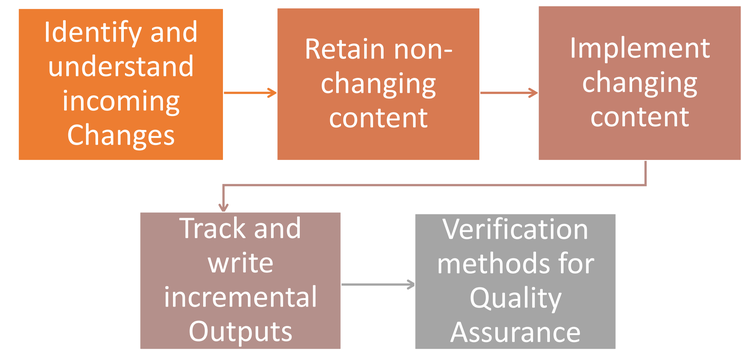
Goal

- ➢ Absorb functional changes
  - ➢ *Functional ECO's* for final milestone intercept
  - ➢ *Derivative programs* with feature refresh
- ➢ Enable physical changes
  - ➢ Intercept late *Pin/Macro movement* feedback
  - ➢ *Block physical dimension changes*

Inputs & Outputs

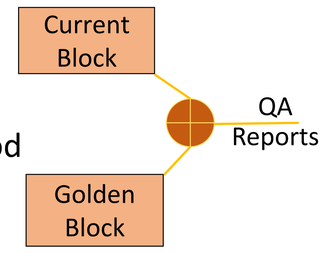


Anatomy of Incremental Framework



Verification Strategy

- ➢ Framework tracks all changes
  - ➢ Incremental collaterals written based on tracked changes
- ➢ XOR used as a verification method
  - ➢ Not used to identify changes as XOR cannot distinguish between expected and unexpected changes.



Different Incremental Solutions

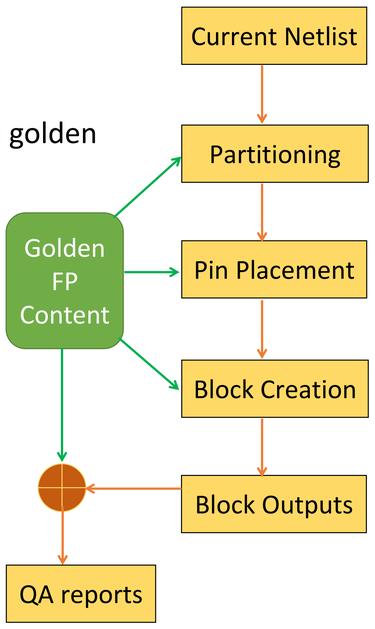
Incremental Translate Mode

Scope

Retain all golden reference content as is with FP re-run

Architecture

Read current netlist → Retain golden content → Write outputs → Verification QA



Usage models

- Porting FP as is between different vendor tools
- Port FP as is and further absorb ECO's for derivative programs
- Port FP as is between different internal environment releases

Benefits

Easy to use, robust QA and enables mixing tools/env

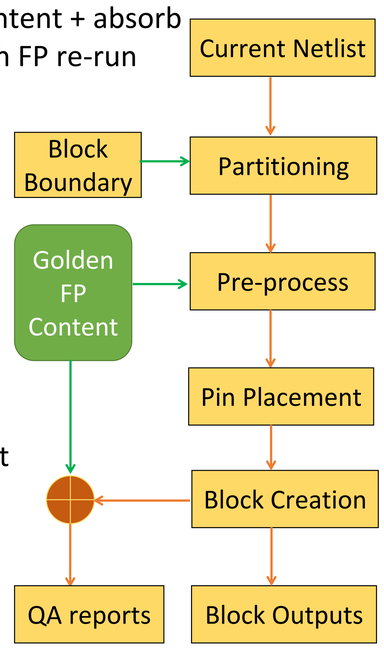
Incremental Retention Mode

Scope

Retain possible reference content + absorb desired physical changes with FP re-run

Architecture

Read current netlist → Read golden content → Process changes → Write outputs → Verification QA



Usage models

- Selective block reuse between different SKU's
- Selective block and parent physical dimension changes (growth/shrink)
- Absorb selective content change (ex-Global clocks)

Benefits

Flexibility to selectively retain and reuse blocks

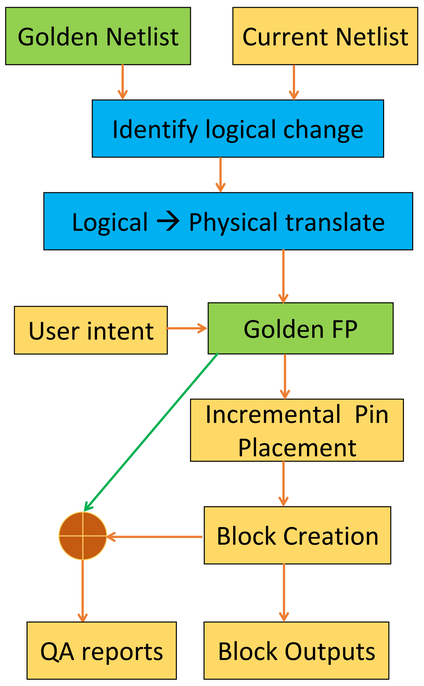
ECO Retention Mode

Scope

Retain possible reference content + absorb minimally invasive physical changes on golden FP

Architecture

Diff netlist → Identify changes → translate changes → Write outputs → Verification QA



Usage models

- Absorb late functional or physical changes
- Selective changes within block boundary – Macro movement, VA resizing

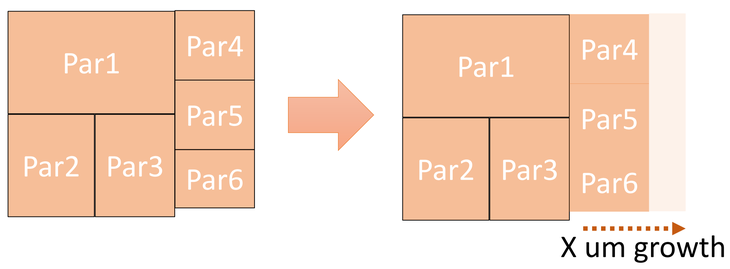
Benefits

Absorb functional changes with minimal perturbation

Comparison and Results

List of items	Incremental Translate Mode	Incremental Retention Mode	ECO Retention Mode
Retention intelligence	✓	✓	✓
QA reports	✓	✓	✓
Pin movements	✓	✓	✓
Macro movements	✗	✓	✓
VA Resizing	✗	✓	✓
Boundary changes	✗	✓	✗
Global clock changes	✗	✓	✗

Incremental Framework Results



Design Reuse scenario

- Design #1 is a hierarchical MIM with 15 blocks
- 3 blocks need to absorb feature for a different SKU→ X growth
- Hierarchical MIM → X growth as a result

- ✓ **12 Blocks reused** as is.
- ✓ 3 blocks topology retained
- ✓ **8X** resource efficiency
- ✓ **80%** timing convergence effort reduction

ECO Retention Results

Metric	Design #1 (Discrete SOC)	Design #2 (~5M GC hierarchy)
Total pin interface changes	11K	40K
Macro changes	~5 in 2-3 blocks	NA
Hierarchy – TAT	40-50% faster	60% faster (4.5 hours → 2 hours)
ECO blocks/Reset blocks	50% (10/20)	60% (3/5)
Block - tool TAT time	2-3 weeks per block	2-3 weeks per block

- ✓ **1-2 weeks** block tool TAT time for each non reset block
- ✓ **100%** of non-changing content retained as is.
- ✓ **1.5X faster** than full design planning flow