Applications:
Hard-IP Reuse & Optimization

June 25, 1999

Session Overview

- Migration tool contents
  - what is needed on top of basic algorithms of Part II?
  - focus on applications
- Case studies
  - applying migration to various real-life designs
  - consequences for design flow and verification
- Hard-IP optimization
  - flow for device sizing and compaction

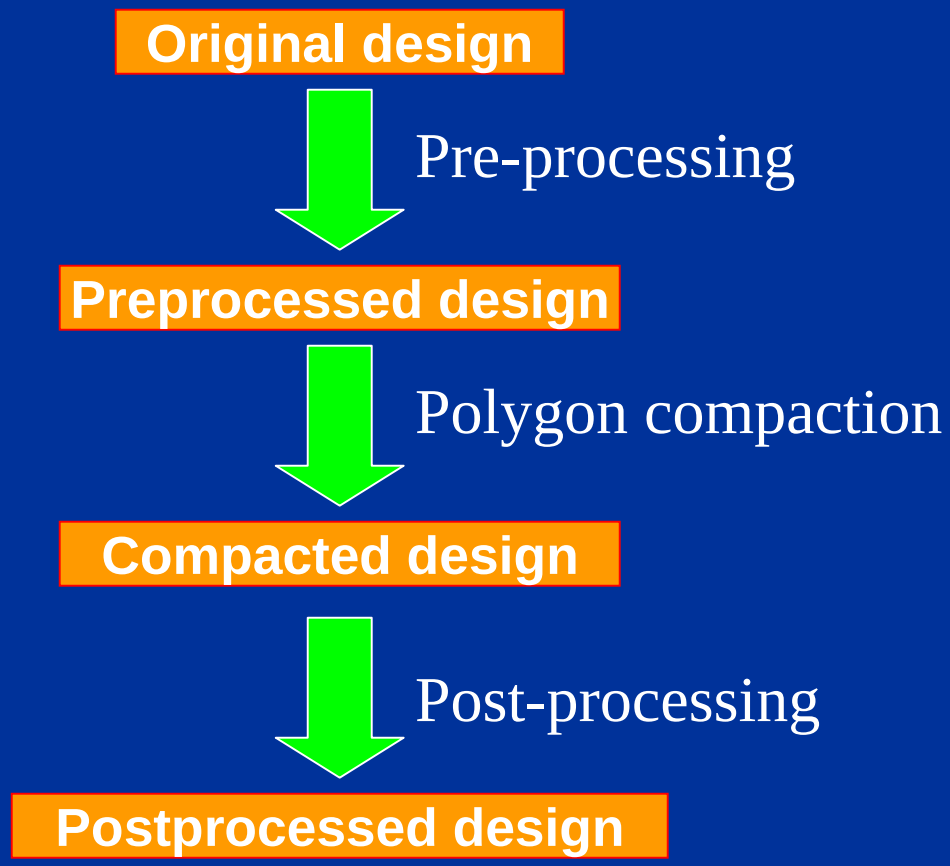
Motivations

- Time To Market
- Reliability
- Maximize leveraging effect of IP Reuse
- Increase speed, reduce power
- Manufacturability

Session Overview

- Migration tool contents
  - what is needed on top of basic algorithms of Part II?
  - focus on applications
- Case studies
  - applying migration to various real-life designs
  - consequences for design flow and verification
- Hard-IP optimization
  - flow for device sizing and compaction

Tools for Hard-IP Reuse



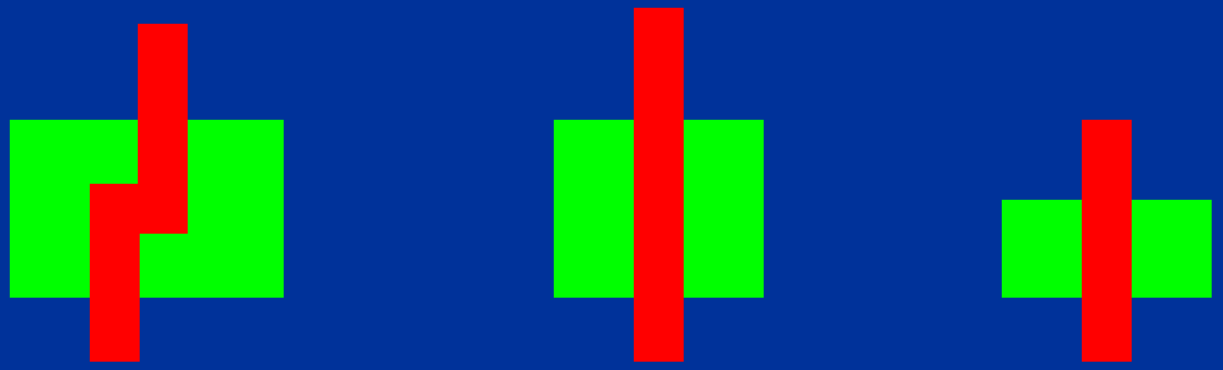
- Device recognition & sizing
- Wide metal control
- Layer arithmetic

- DSM design rules
- Pin gridding
- Device sizing

- Via/contact optimization
- Layer arithmetic

Device Recognition & Resizing 1

- User control over device sizes
  - transistor W and L
  - capacitors
  - resistors
- Prevent compaction to minimum dimensions



Original X-compaction Y-compaction

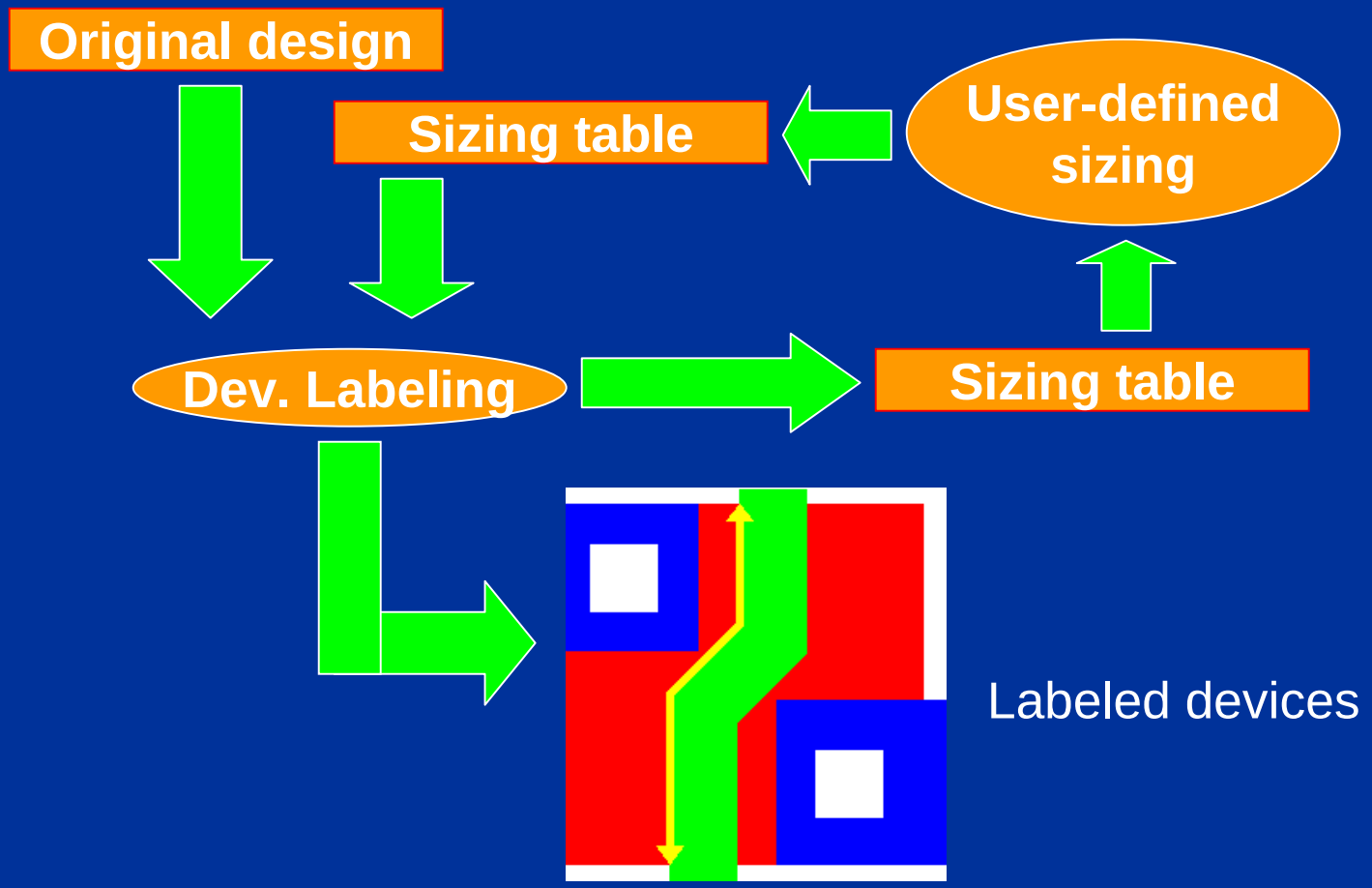
Device Recognition & Resizing 2

- Automatic transistor resizing control
  - W/L resize factors, piece-wise linear, e.g.

W-value	W/L resize factor
0..2.4u	1.0
2.4..4.0u	0.9

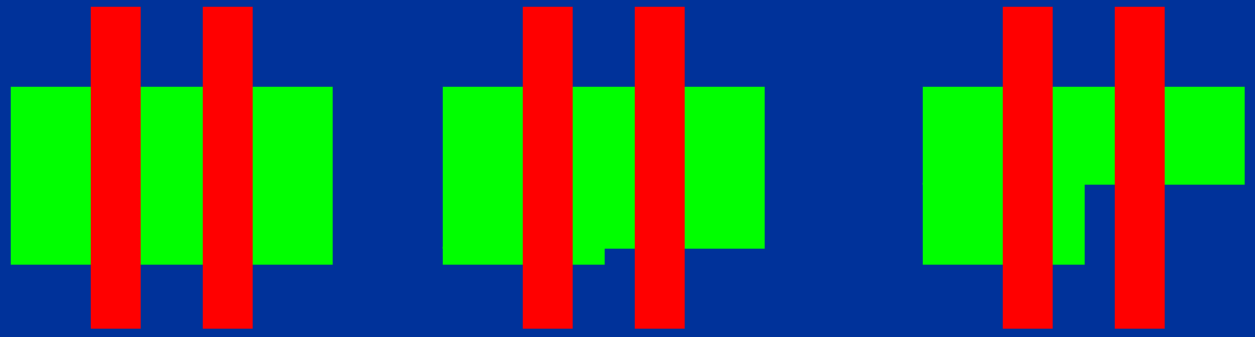
- User defined transistor resizing from
  - text-labels in layout
  - user-specified sizing-table
  - user-created resizing program
- Resizing of capacitors and resistors

Device Recognition & Resizing 3



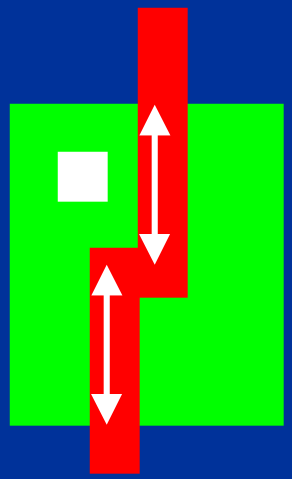
Device Recognition & Resizing 4

- Wire length control in compaction engine to achieve correct W
- Needs support for (45-degree) bent gates, crossing 45-gates.
- Jog insertion to resize gates correctly

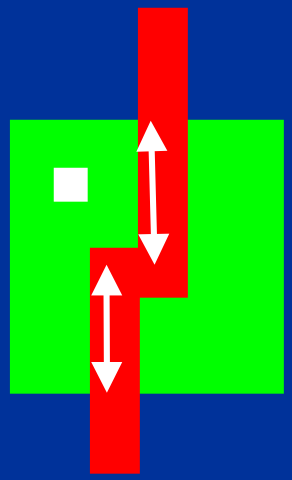


Device Recognition & Resizing 5

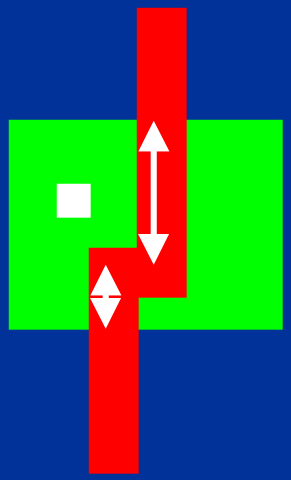
- Length borrowing between gate segments



Original
both segments
same size



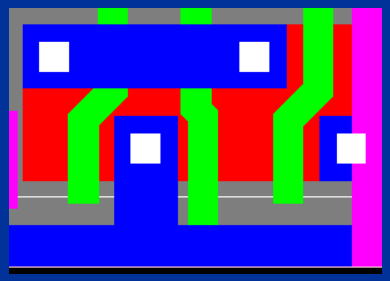
No borrowing
both segments
same size



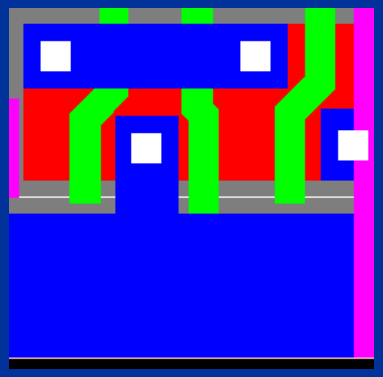
With borrowing
lower segment
smaller

Wide Metal Control 1

- Prevent wide metals from being compacted to minimum dimensions, affects performance and reliability.



No wide metal control

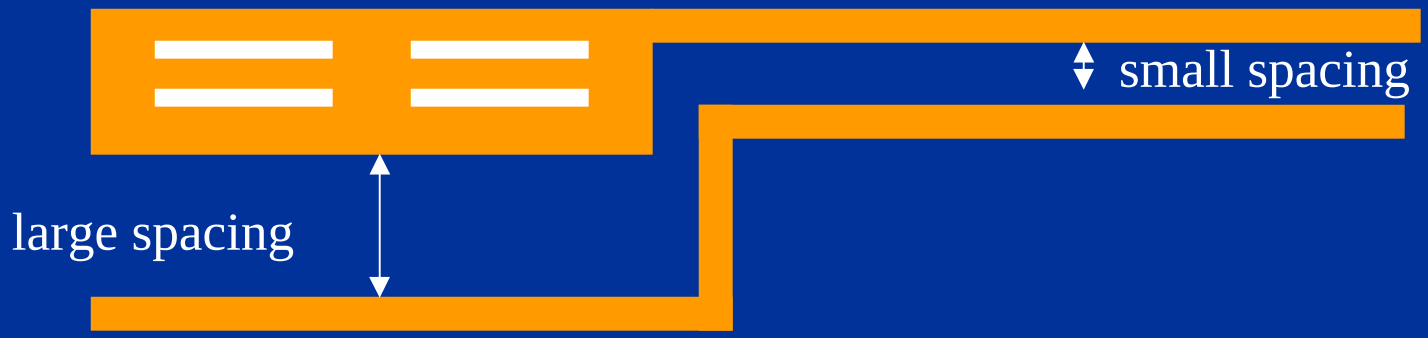


wide metal control

- Recognize wide metals, then label & resize
- Resizing controls include proportional/table-mapping

Wide Metal Control 2

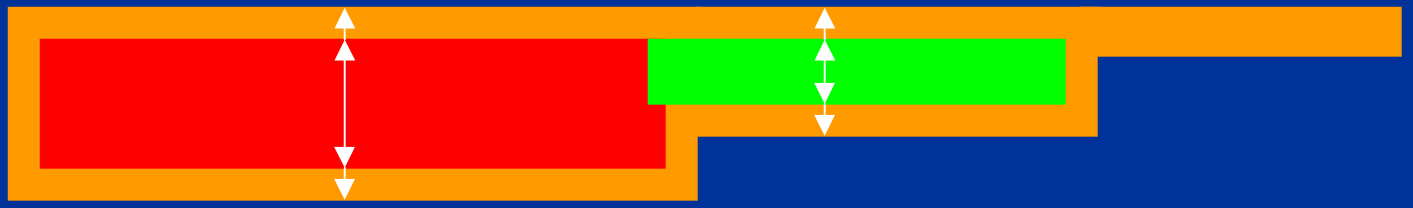
- Dedicated wide-metal spacing rules



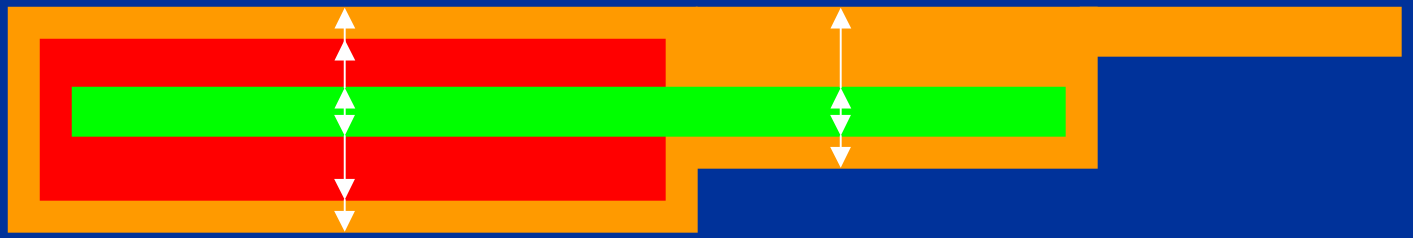
- Implement wide-metal stress-relief rules (slotting)

Wide Metal Control 3

- Example wide-metal handling strategies
  - dedicated layers for each wide-metal:

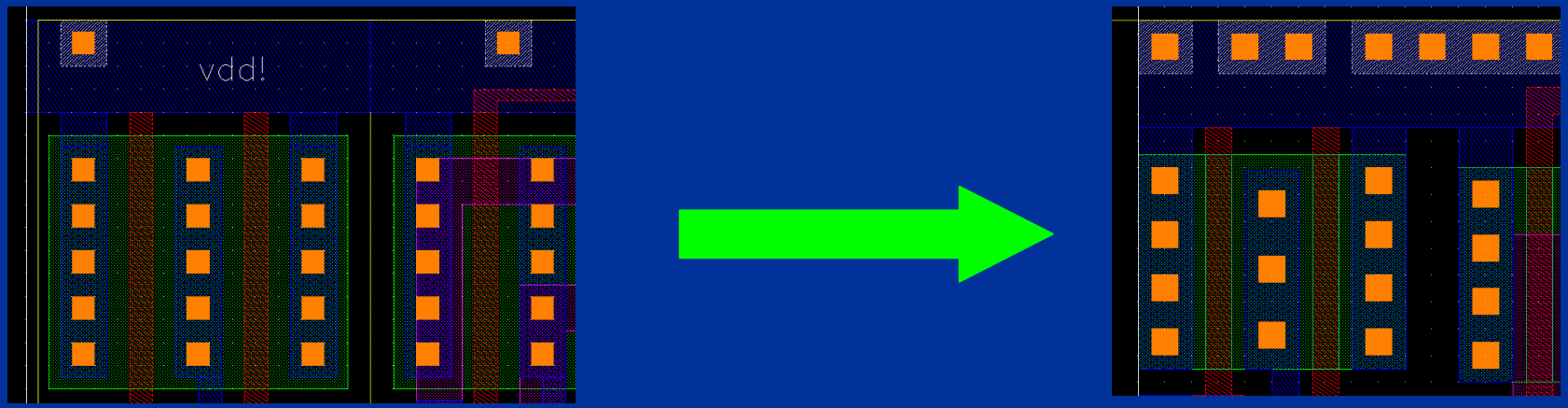


  - nested layers that together implement wide-metal:



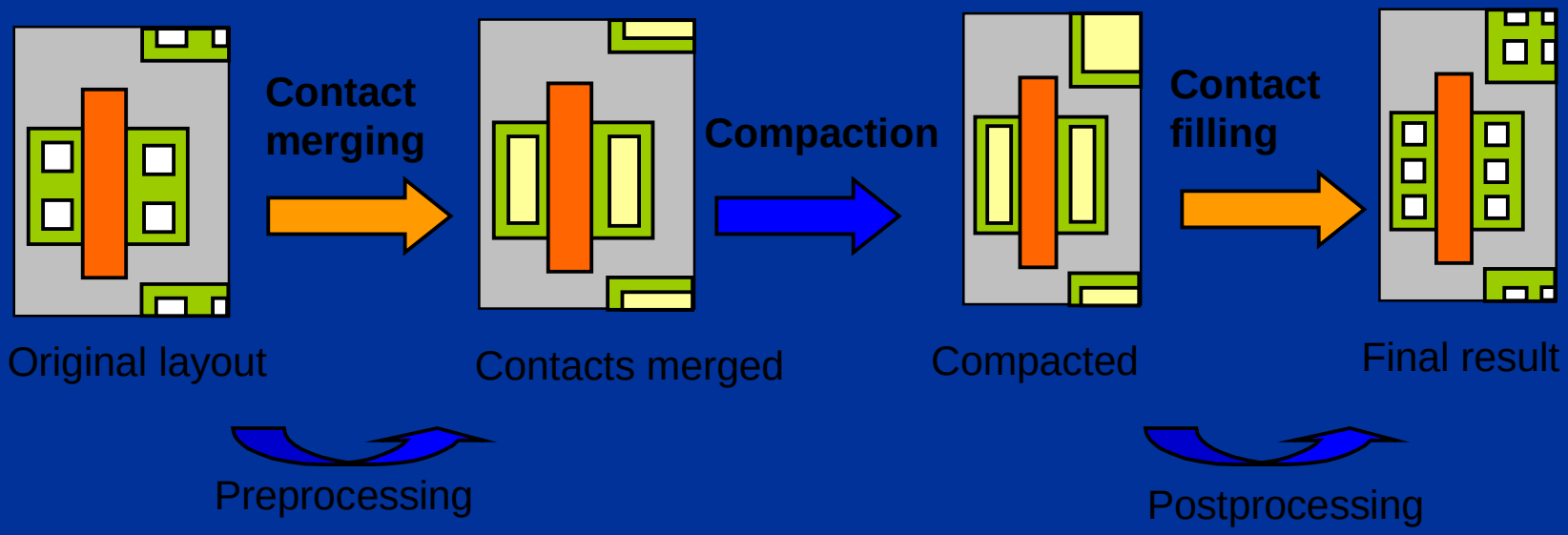
Contact/Via Optimisation 1

- Contacts next to gates
  - avoid gate sizing errors
  - enhance performance, reduce size
  - accommodate for salicide/non-salicide processes



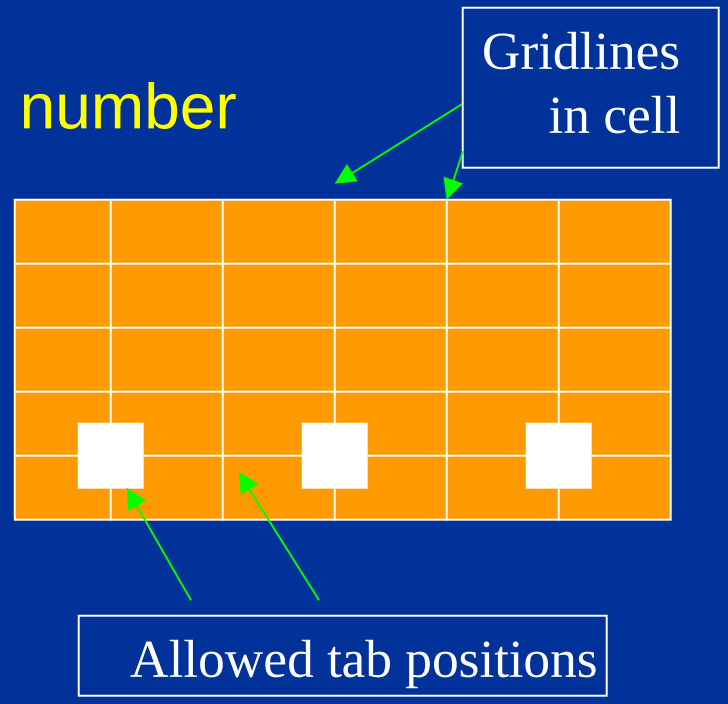
- Well and substrate tabs, vias in routing
  - improving reliability, less susceptible to latchup

Contact/Via Optimisation 2



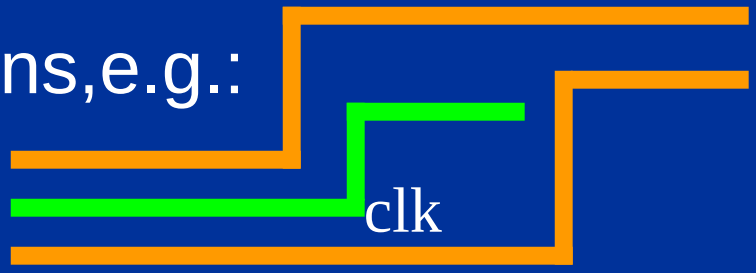
Contact/Via Optimisation 3

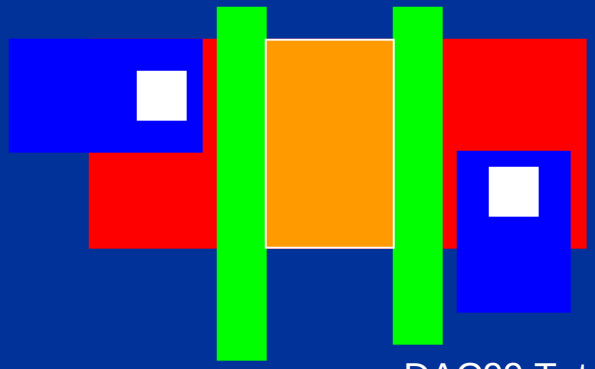
- Controls
  - as much as fit -> WLO
  - reduce to 1
  - growth-direction, maximum number
  - support for arrays
  - gridded filling
- Must also support
  - 2-sides extension rules
  - rectangular contacts



Toolbox 1 : Polygon Arithmetic

- GDS2 -> GDS2
- All boolean mask operations
- Hierarchical operation
- Netlist dependent operations,e.g.:
  - all polygons on net "clk":

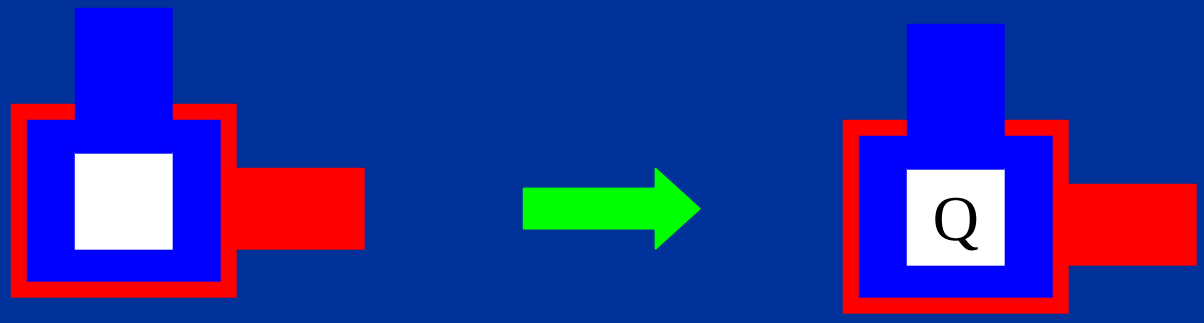




- all source/drain areas without a contact

Toolbox 2 : Layout Manipulation Goodies

- Selective layer/structure flattening
- Grid-line insertion
- Text-label insertion:

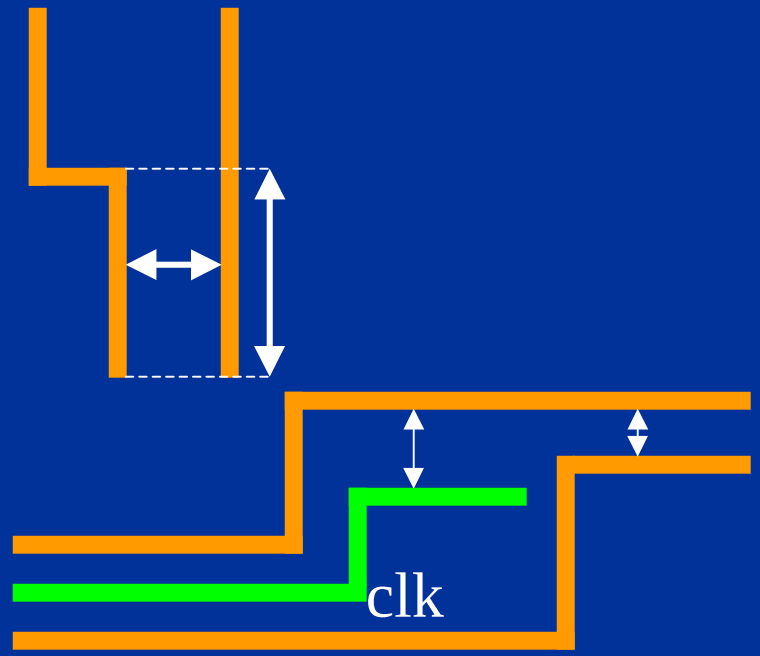


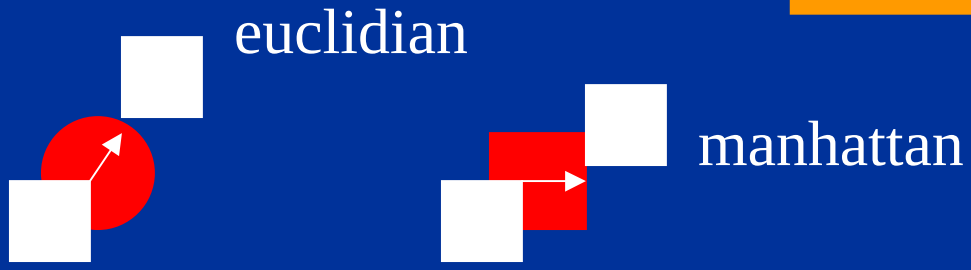
Compaction 1: Basics

- Designrule-scanning
- Constraint graph solving, overconstraint resolution & reporting
- Wire length minimization
- 45-degree polygon edges
- Engine characteristics
  - 1/2 dimensional
  - flat
  - hierarchical

Compaction 2: context-dependent design rules

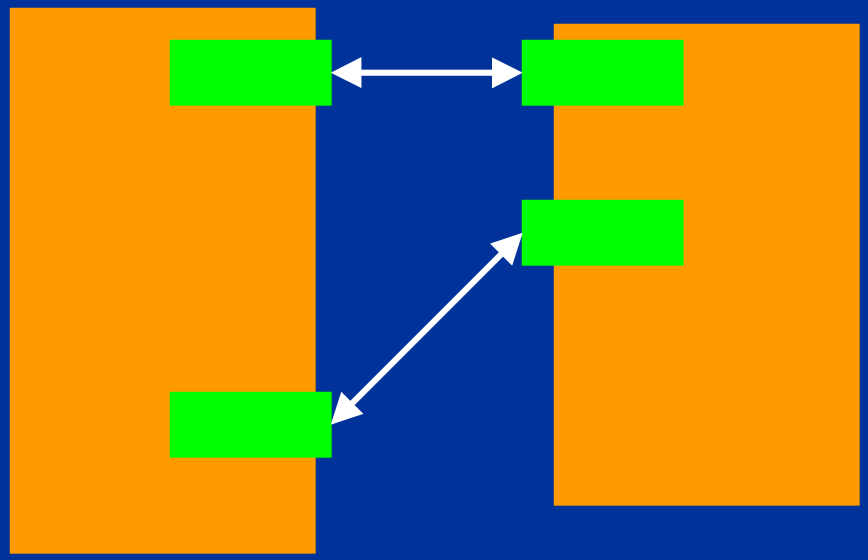
- Common run-length
- Connectivity-based:
  - selection of certain signals
  - layer present on net
  - same-net/different net
- Corner rule values





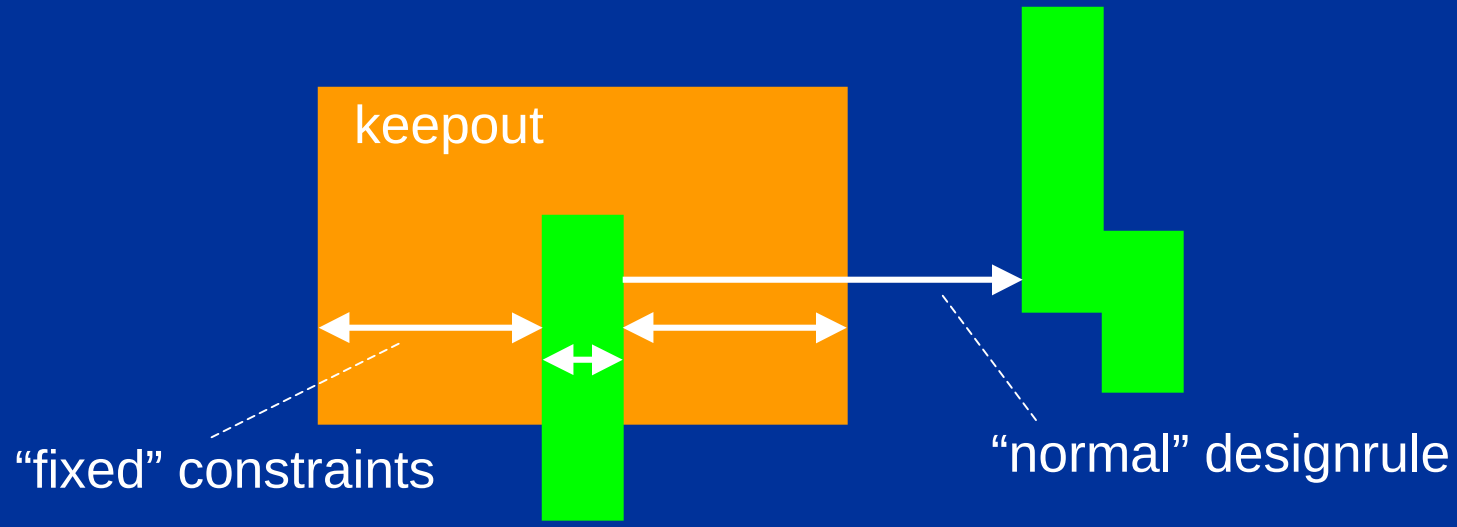
Compaction 3: abutment concept

- Align polygons in different cells during compaction
- Used for: large clocks, cell libraries, memories



Compaction 4 : Keepout areas

- Needed to identify areas as "don't touch"



- Need to be programmable to the end-user:
  - Keepouts for “only poly and active”
  - Keepouts for “everything but metal1”

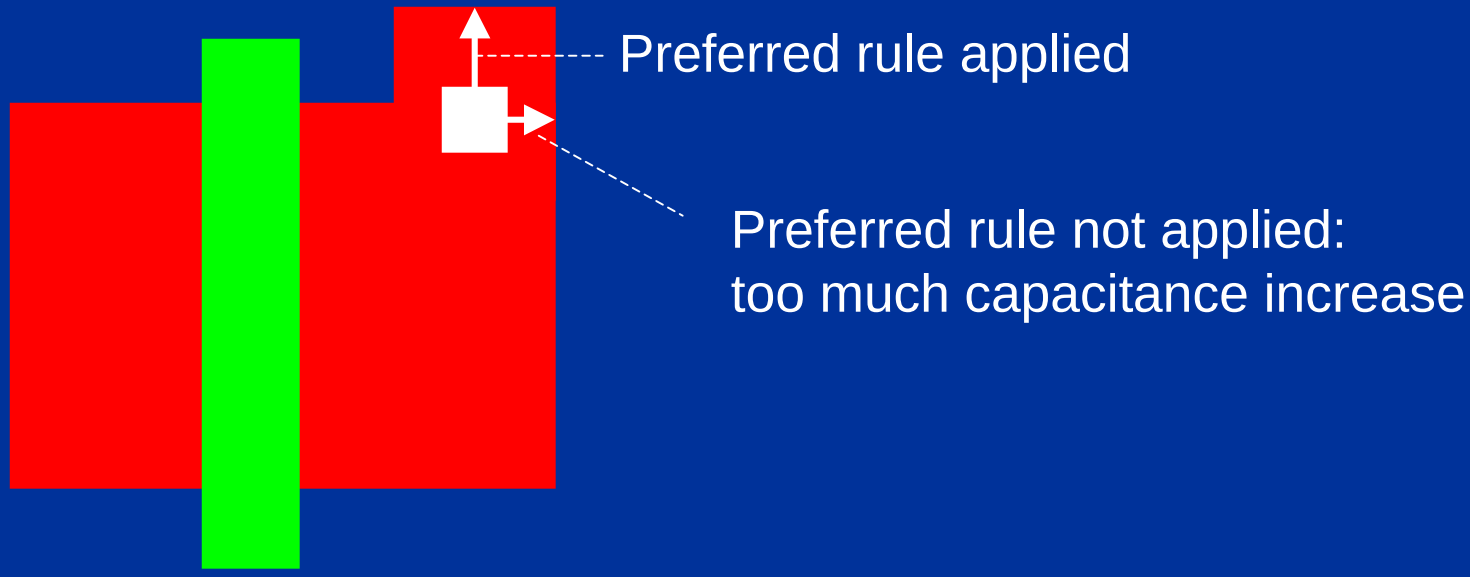
Compaction 5 : Examples of DSM rules

- Preferred values
- Minimum area requirements
- 2-sides overhang rules:

- Polysilicon hammerheads
- Forbidden regions, e.g.:
  - Spacing = 0.5 micron or >= 1.2 micron
- Larger spacing rules for “small” wide metals

Compaction 6: Preferred rules

- Preferred rule-values need priorities:
  - Discriminate yield effect between different preferred designrules, slack can be used only once.
  - Trade-off with performance of cell:

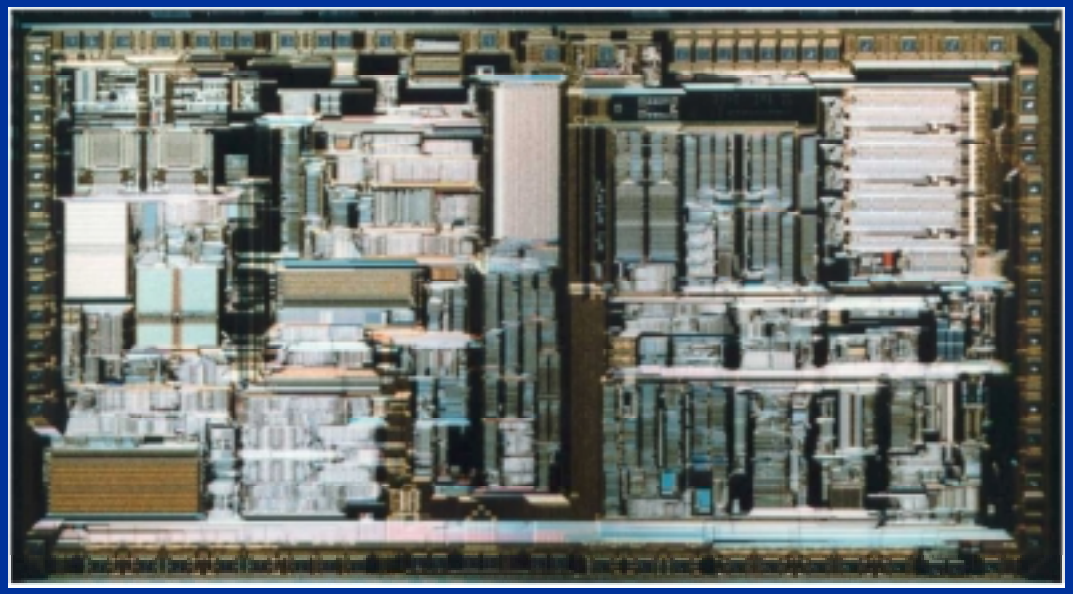


Session Overview

- Migration tool contents
  - what is needed on top of basic algorithms of Part II?
  - focus on applications
- Case studies
  - applying migration to various real-life designs
  - consequences for design flow and verification
- Hard-IP optimization
  - flow for device sizing and compaction

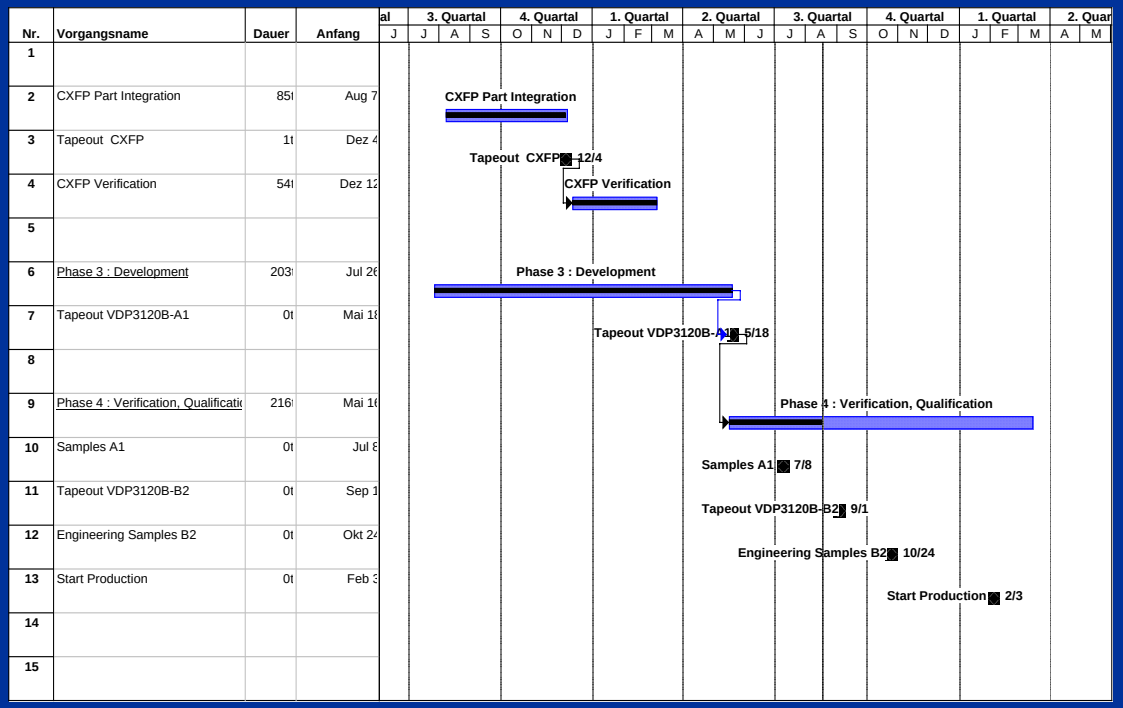
Case study 1 : ITT Intermetall single chip video

- 2 chips merged into 1, 0.8 to 0.5 micron
  - 800K transistors
  - full-custom
  - 80% digital
  - 20% analog
  - RISC controller
  - DSP
  - RAM, ROM, FIFO
  - A/D, D/A converters, clock oscillator



Case study 1 : Migration effort

- 5 manyears
- 14 months elapsed
- 1 test-chip
- first-time-right silicon



Case study 1 : Digital part migration

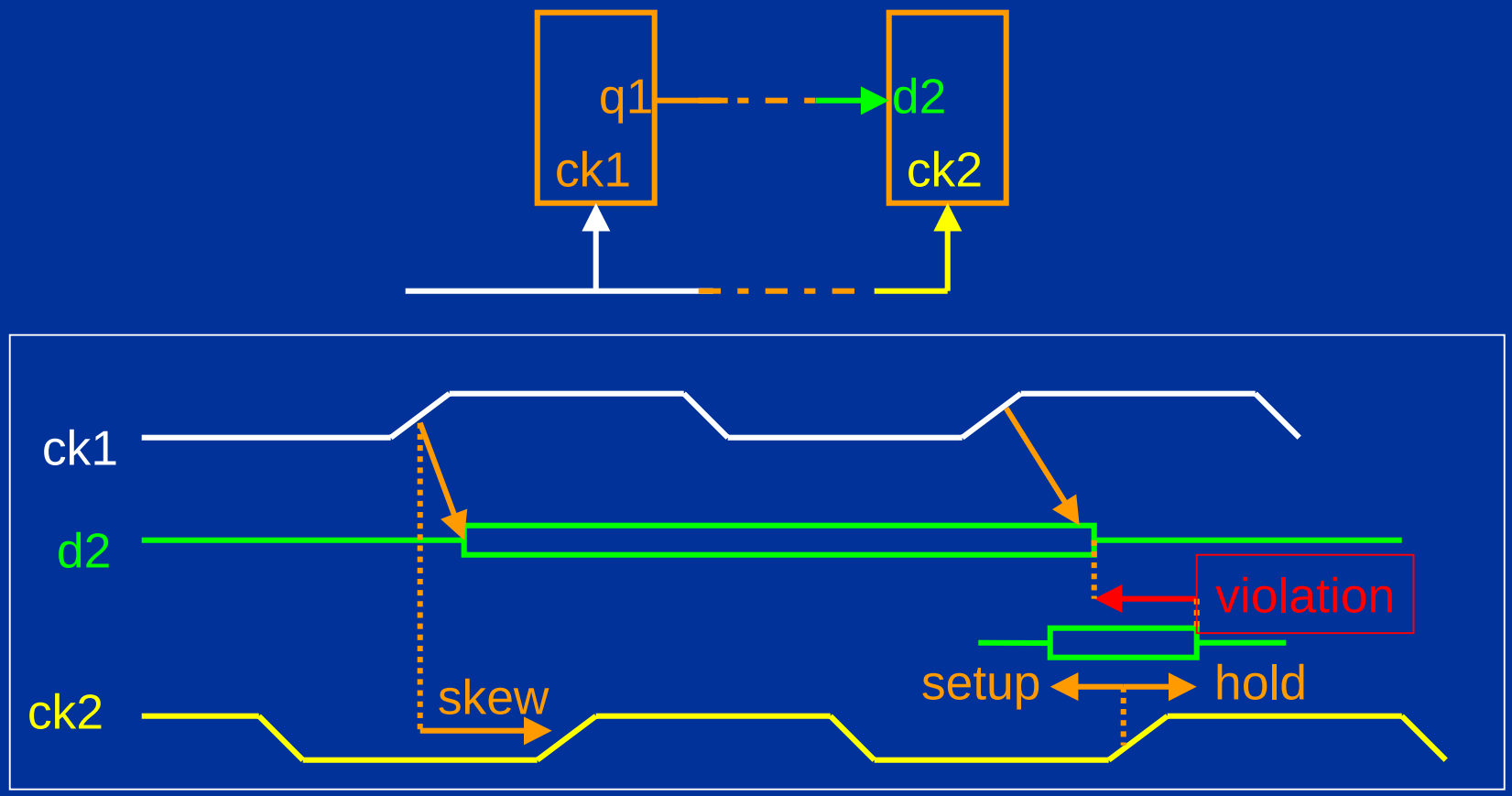
- Each block compacted separately
- Automatic gate-resizing
- Wide metals preserved, scale with technology
- Blocks can be handled by different teams

Case study 1 : Digital part verification

- SPICE simulations on resized cells, determine technology W/L ratios
- Extraction and SPICE simulations of critical nets (clock)
- No timing characterizations on complete blocks or designs!
- Test-chip half-way to verify approach

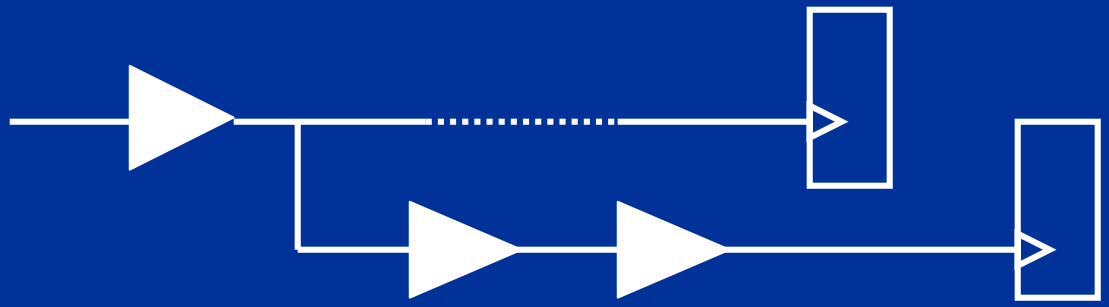
Case study 1 : clock skew causes 1

- Clock skew may cause hold time violations



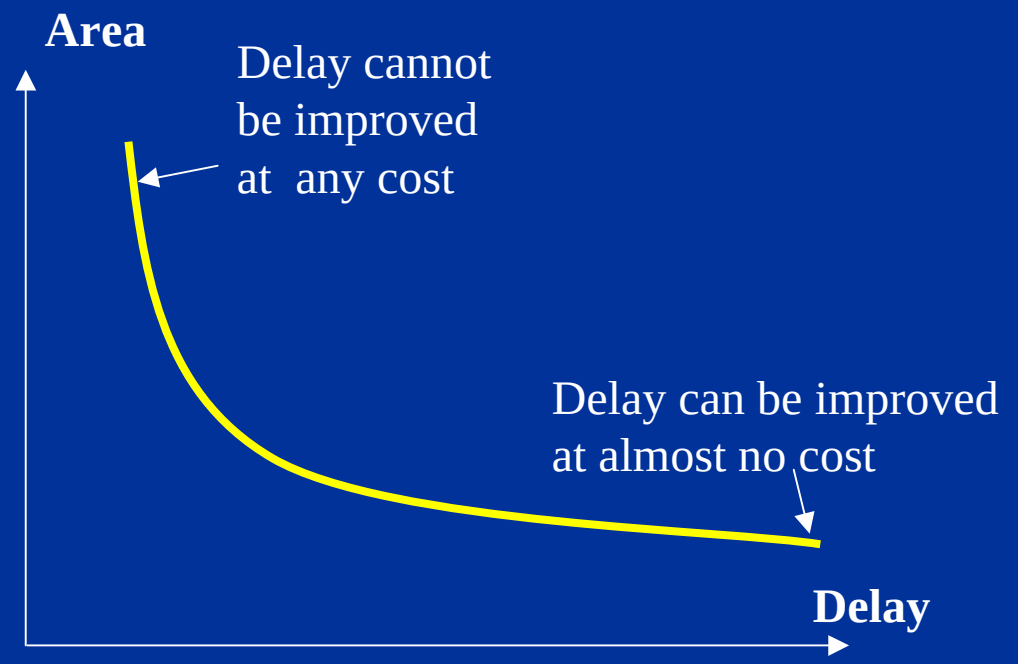
Case study 1 : clock skew causes 2

- Clock skew caused may be caused by:
  - excessive wire length
    - note: migration doesn't change topology
  - insufficient drive strength clock buffers
    - enforce larger buffers
  - unbalanced clock trees:



Case study 1 : migration critical design styles

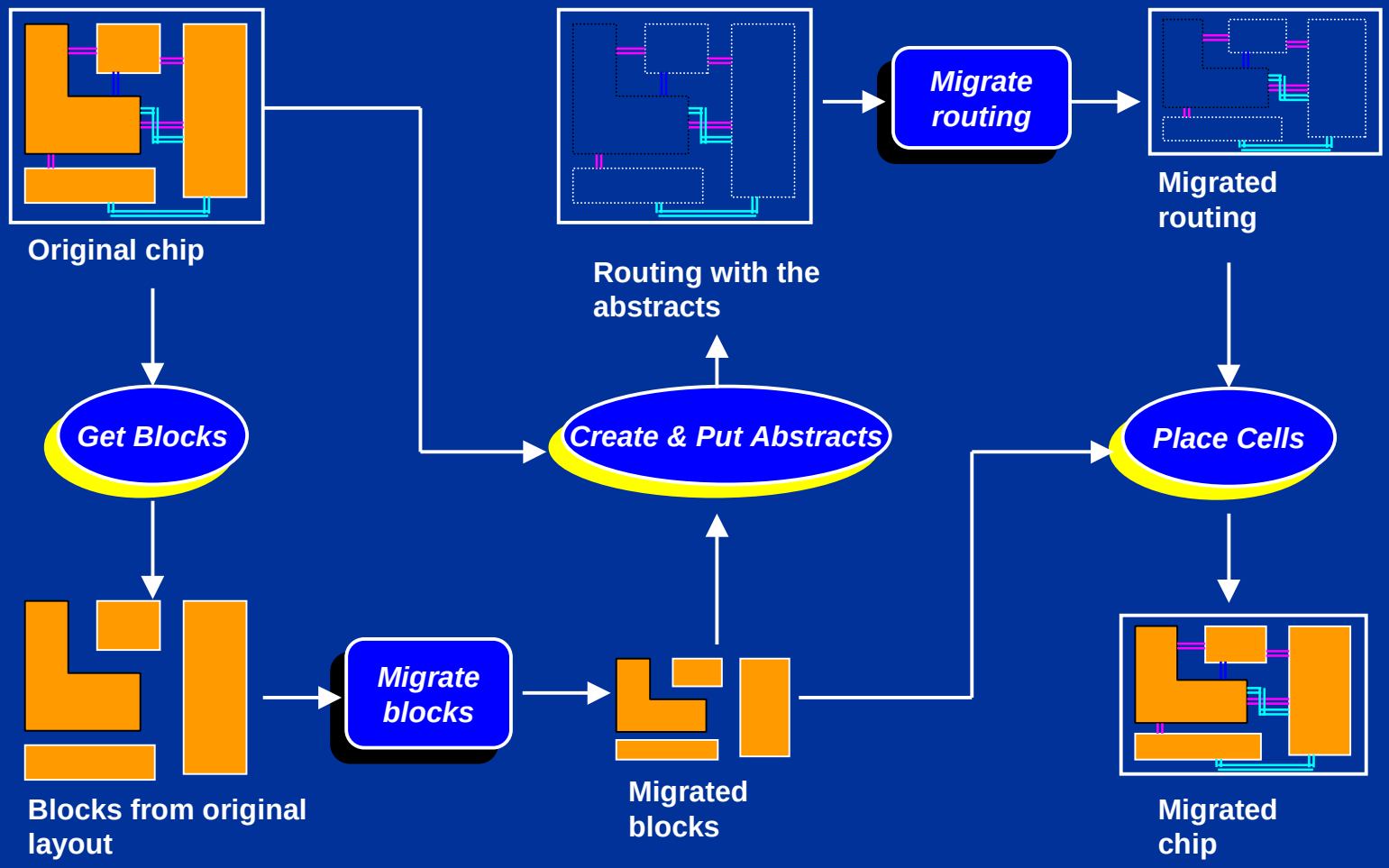
- Gated clocks
- Unbalanced clock trees
- Pre-charged logic, sensitive to crosstalk
- Asynchronous logic
- Outer edges of area/delay curve



Case study 1 : Migration of analog

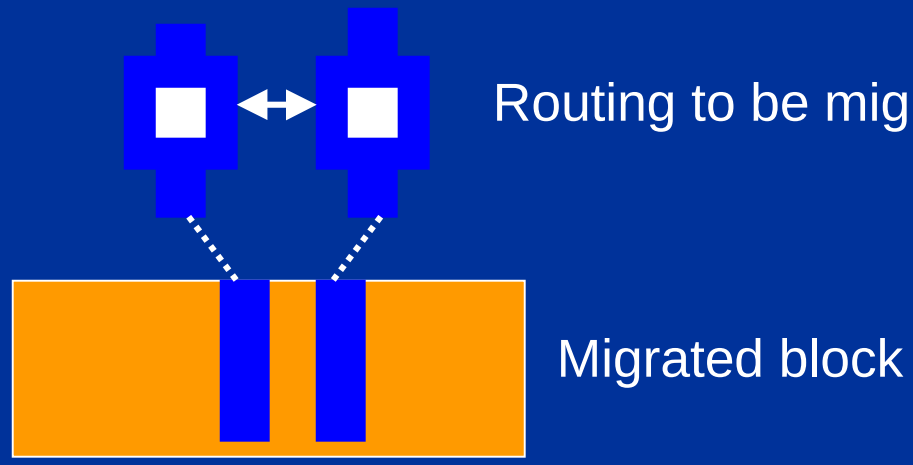
- Shrink and regrid concept: each new coordinate snapped to nearest process grid
- Take care of 45-degree angles
- Verification through :
  - extraction
  - SPICE simulation
  - manual layout changes

Case study 1 : Migration of global routing 1



Case study 1 : migration of global routing 2

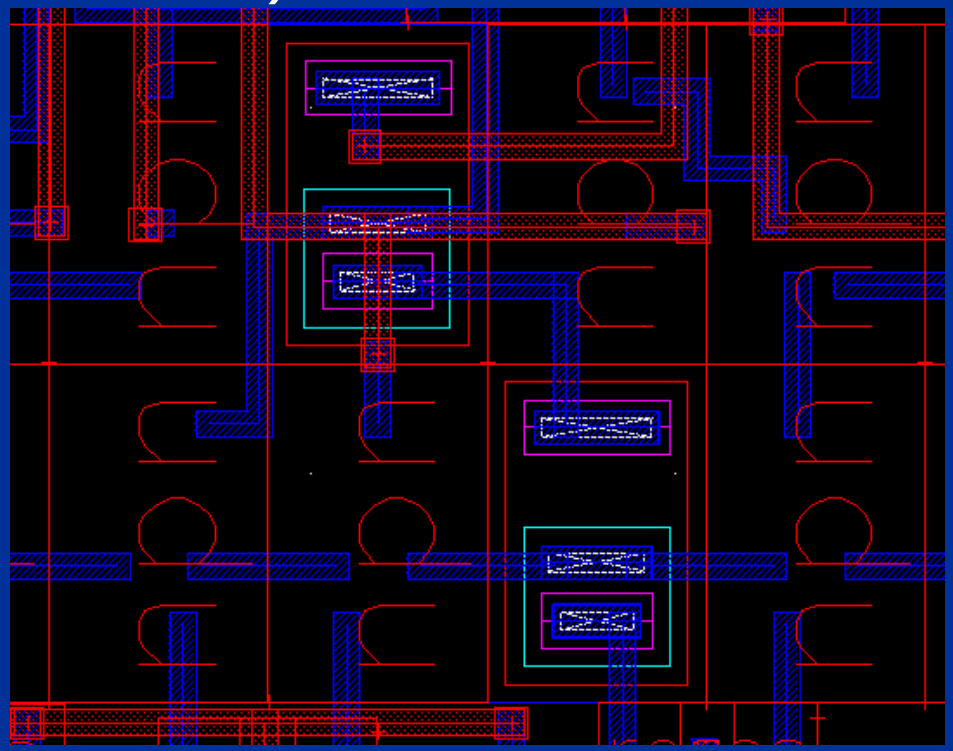
- Migrate routing such that it reconnects to pins of migrated blocks
- Danger : pin positions in block too close for routing:



- Solution: ring-of-routing when migrating blocks

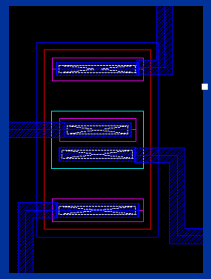
Case study 2 : Cell based analog 1

- Aim is to migrate or optimize cells and routing
- Result must have (Pcell-based) instances
- Preserving hierarchy :
  - ECO's
  - verification

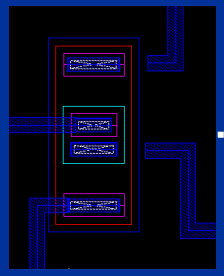


Analog compaction 1: device substitution flow

- Resizing polygons does not work!

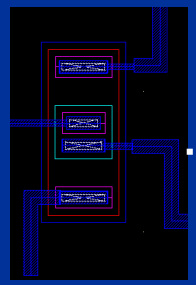


Original design



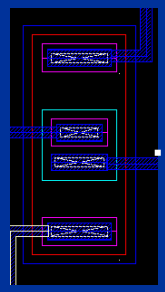
Substitute devices

Substituted design



Re-connect terminals

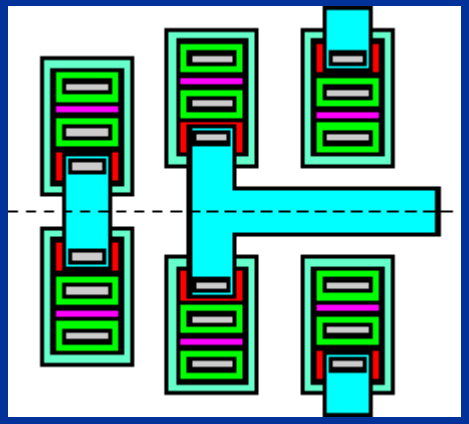
LVS-correct design



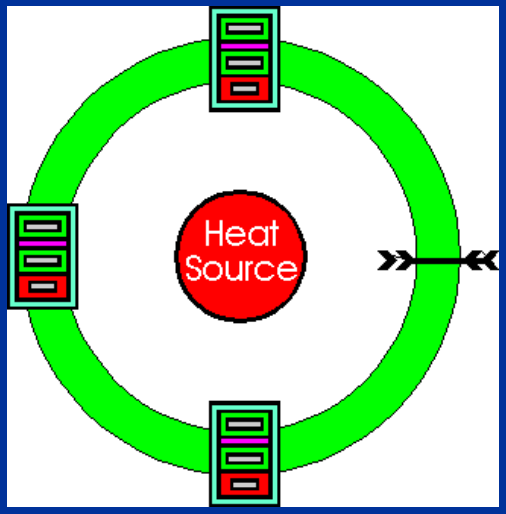
Hierarchy preserving compaction

Migrated design

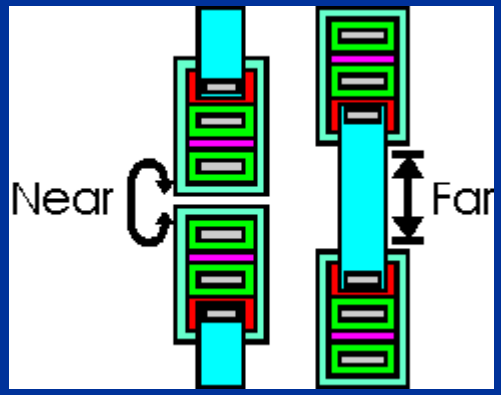
Analog compaction 2: analog constraints



Symmetry



Isothermic regions



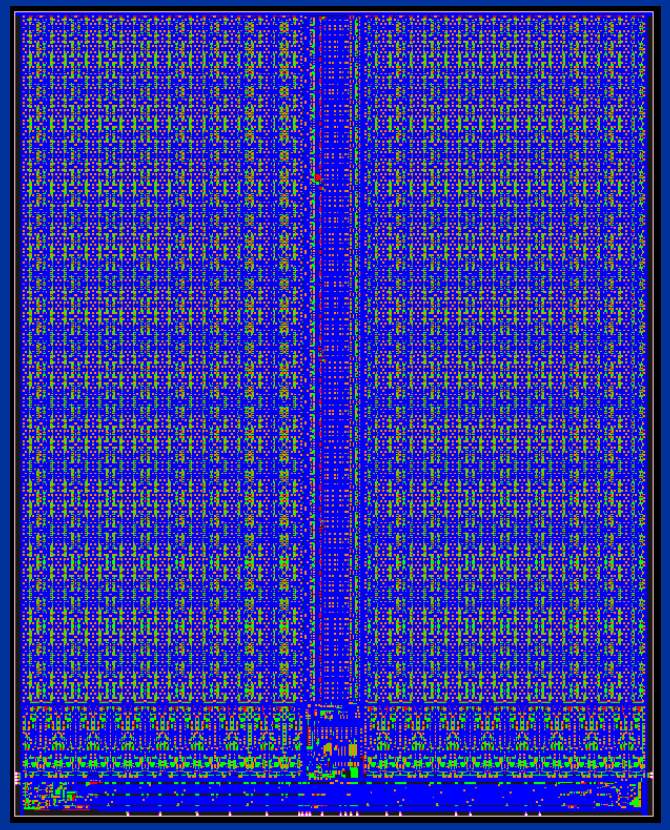
Near/far constraints

Case study 2 : hierarchy preserving compaction

- Hierarchy preserving compaction
  - instances kept as is, do not change
  - only polygons in top-level (routing) are compacted
  - instances allowed to move
  - implementation may use keepout areas around instances
- This differs from hierarchical compaction:
  - hierarchical compaction modifies all hierarchy levels at the same time.

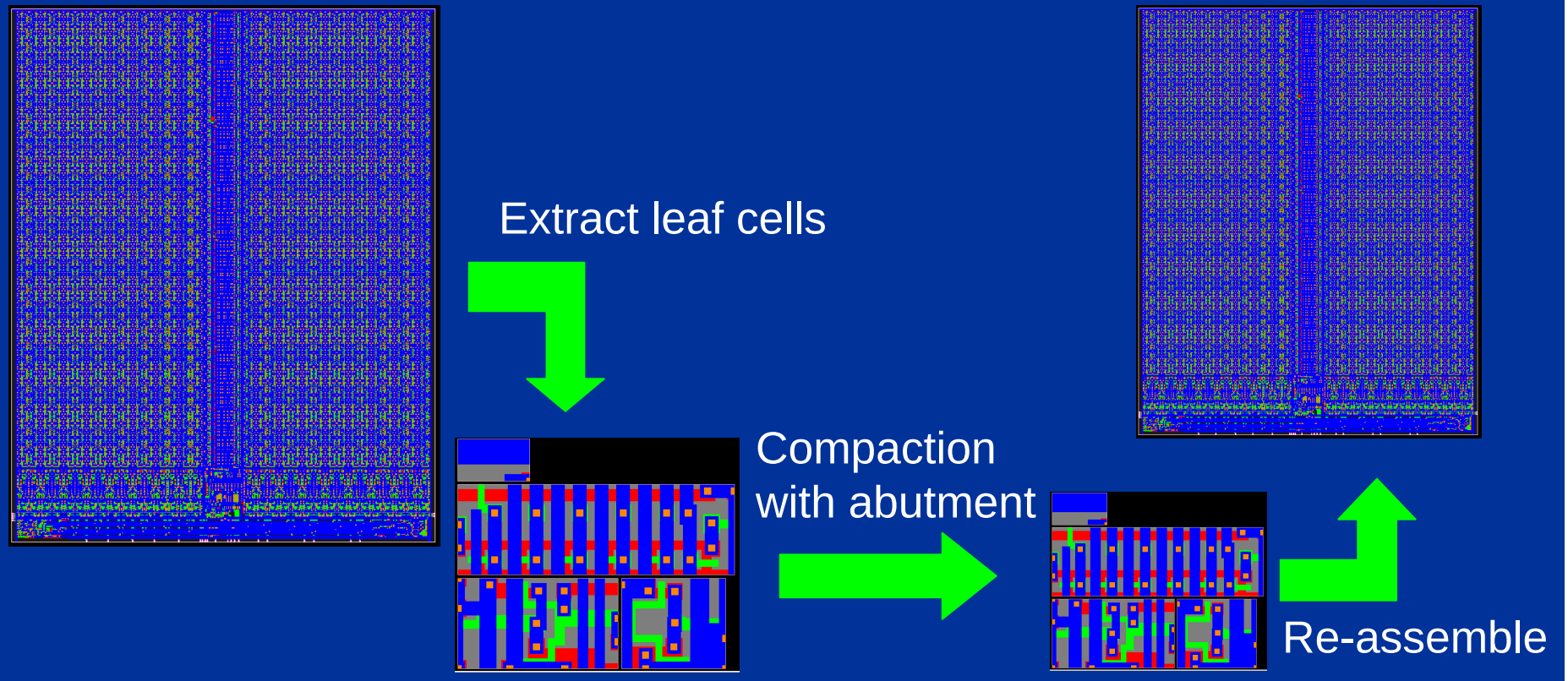
Case study 3: memory instance

- Characteristics
  - full-custom designed
  - very hierarchical (overlays, non-matrix)
  - highly regular
- Requirements
  - core cell must be identical
  - keep hierarchy: verification and database size



Case study 3: Flat compaction

- Exploit regularity of design



Case study 3: Hierarchical compaction

- Memory instance given to compactor as is
- No pre/post processing needed
- Compactor analyses cells for overlays and inter-cell rules
- Applicable to memory generators:
  - input is a collection of representative instances
  - each cell must occur once with each of it's possible neighbours in hierarchy

Case study 3: Verification

- Hierarchical compaction eases verification:
  - same cells, same hierarchy
  - hierarchical DRC and LVS can be used
- For flat (tiled) approach LVS and simulation setup needs to be modified
  - different set of leaf-cells
  - over-the-cell routing is now part of leaf-cells

Designs unsuited for Hard-IP reuse

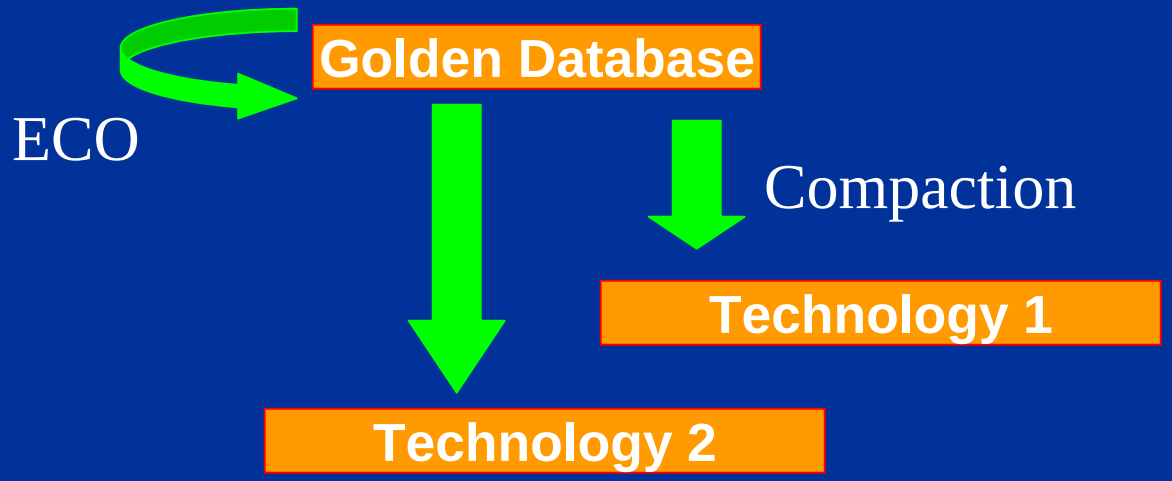
- Large synthesized layouts, no manual tweaks
- Need to use more metal layers:
  - compactor cannot exploit additional metal layers
  - result and original have same number of metals
- Constraints on design size
- Process dependent designs
  - many analog designs
  - DRAM core-cells

Hierarchical-flat compaction comparison

Flat	Hierarchical
large database	small database
difficult ECO's	ECO's doable
optimal density	area is larger
different verification	same verification flow
large blocks possible	size limits!
each device uniquely sized	limited sizing possibilities

Golden database compaction

- Golden hierarchical database is compacted to different technologies
- ECO's always in original database



Verification techniques summary 1

- Don't:
  - analyse &verify critical nets, determinate technology W/L ratio, digital only, requires “neat” design style
- Extraction and simulation
  - for memories, analog, digital
  - requires large capacity switch level simulator
- Transistor level static timing analysis
  - digital only
  - tools recognize flipflops
  - quick & reliable: if critical path meets spec., it's ok

Verification techniques summary 2

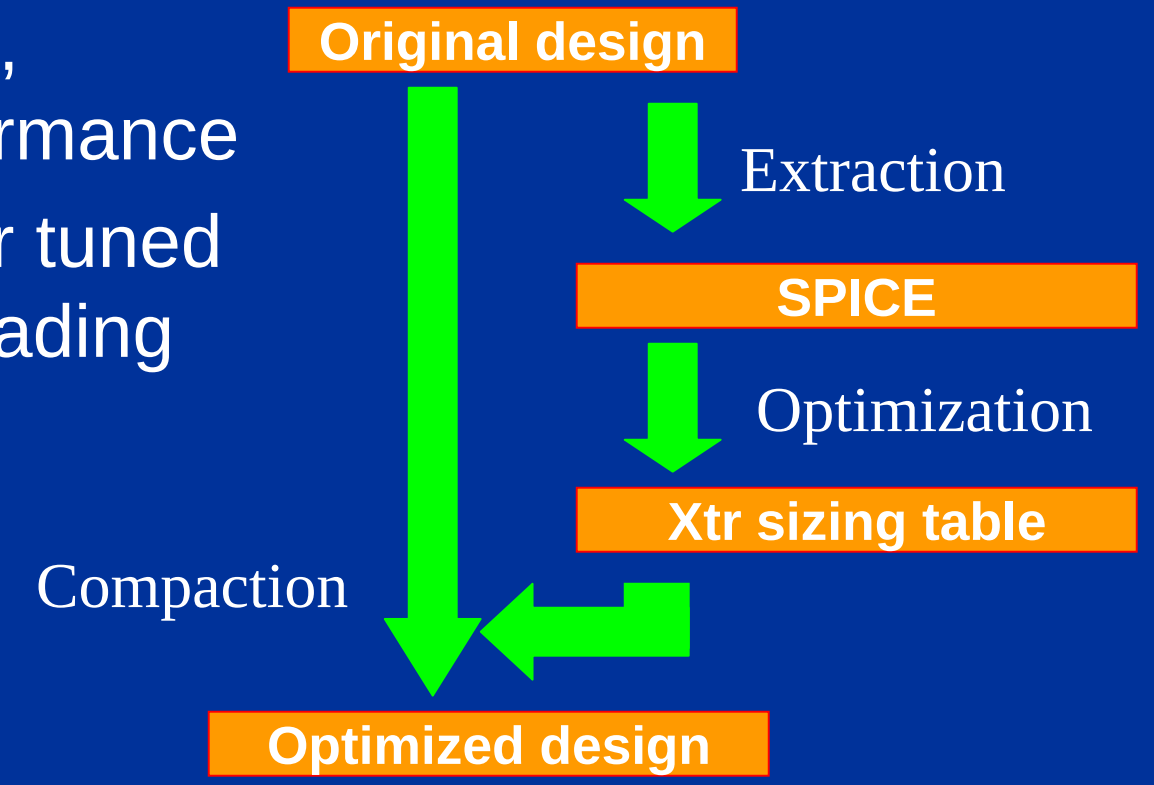
- Gate level timing models and routing backannotation
  - needs hierarchical compaction or
  - cell characterization on flat layouts:
    - cells recognized from extracted transistor netlist
    - “similar” cells are grouped and characterized, only for their local environment
    - routing extracted

Session Overview

- Migration tool contents
  - what is needed on top of basic algorithms of Part II?
  - focus on applications
- Case studies
  - applying migration to various real-life designs
  - consequences for design flow and verification
- Hard-IP optimization
  - flow for device sizing and compaction

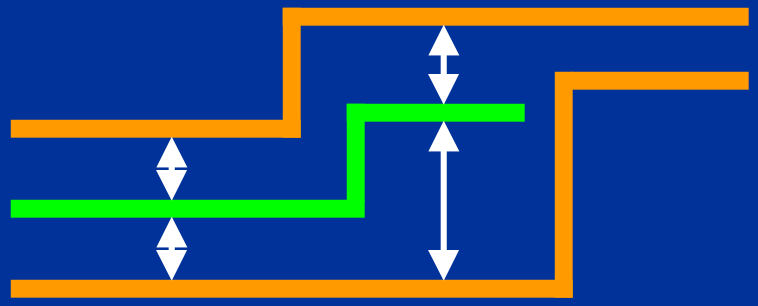
Hard IP Optimization

- Links migration engine to device sizing tools
- Reduce power, increase performance
- Each transistor tuned for it's exact loading



Hard IP Optimization : interfaces

- Simple interface based on sizing table
- Devices need to be uniquely identified in layout (fingered transistors!)
- Device optimizer reads SPICE produced by extraction tool
- Cross-coupling part of optimization -> polygon-based constraints



Hard IP Optimization : Full-custom designs, no hierarchy

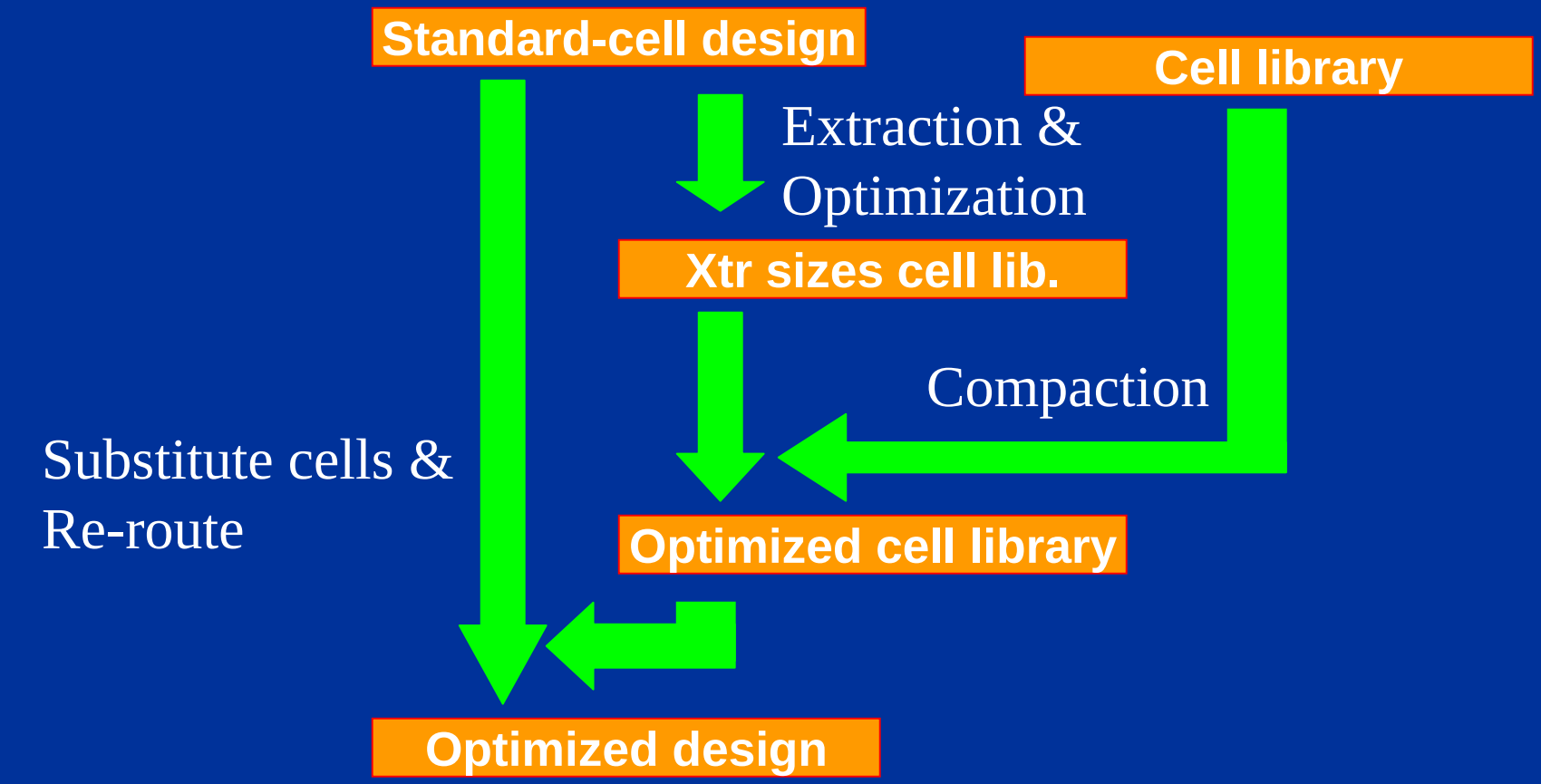
- Example designs: datapath, custom control logic
- Maximum benefit: each device individually customized
- Use flat compaction
- Timing verification by same static timing analyzer underlying device optimization

Hard IP Optimization : Full-custom designs, keep hierarchy

- Example designs: datapath, memory
- Use hierarchical compaction
- Device optimization limited by identity constraints
- Suggestions to user for copying leaf-cells

Hard IP Optimization : Semi-custom

- Blocks too large to be compacted flat or hierarchical



Conclusions

- Total solution: device sizing, migration engine and transistor level static timing analysis
- Fast time-to-market
- Designs optimized for performance/power/area/signal integrity/DfM