Modern Physical Design: Algorithm Technology Methodology (Part III)

Stan Chow Ammocore

Andrew B. Kahng UCSD

Majid Sarrafzadeh UCLA

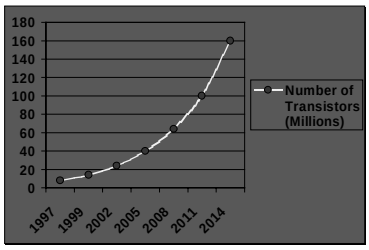
Goals of the Presentation

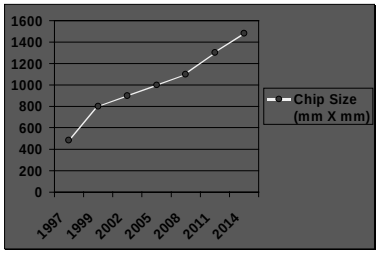
- Various methodologies and interaction between sub-tools
- Classification of Techniques

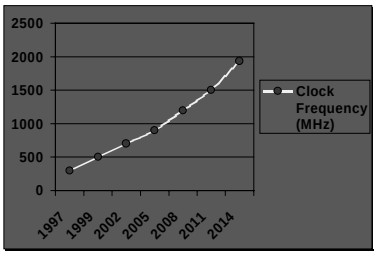
PART III: Interaction with Upstream Floorplanning and Logic Synthesis

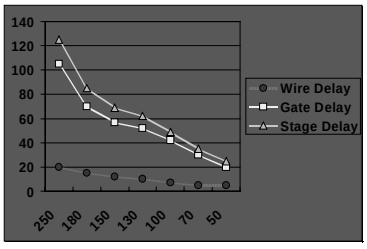
- Interaction classification
  - Definitions
  - Pros and Cons

Design Trend



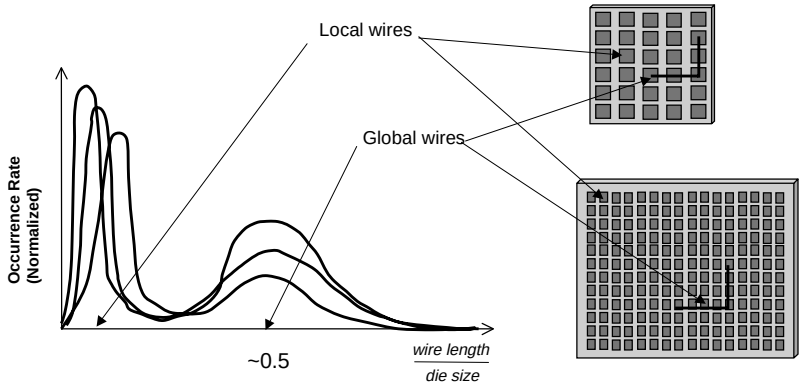




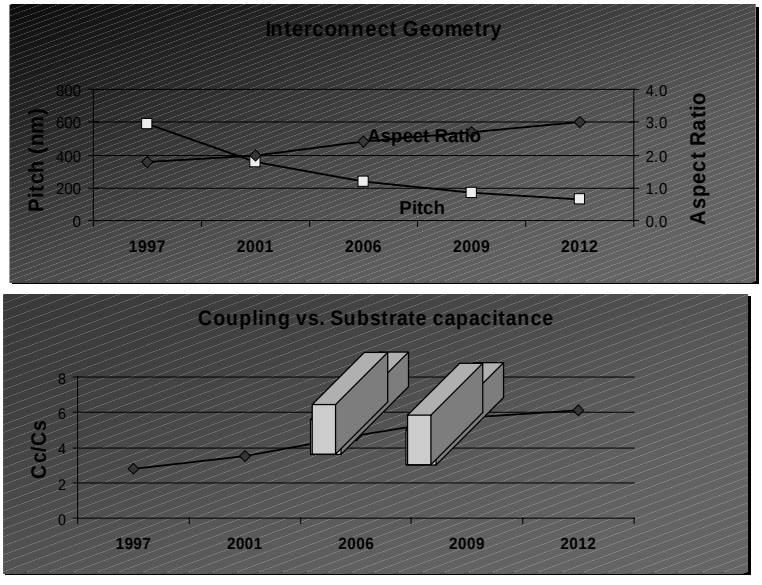


Source: 1998 Update, International Technology Roadmap for Semiconductors, SIA in co-operation with EECA, KSIA, EIAJ, TSIA

DSM: Design Global Wires



DSM: Crosstalk

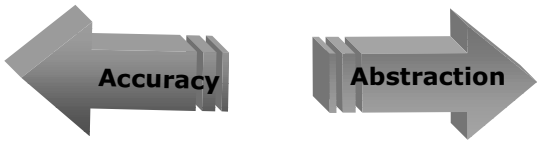


Source: 1998 Update, International Technology Roadmap for Semiconductors, SIA in co-operation with EECA, KSIA, EIAJ, TSIA

Observation 1

- Deep Sub-micron (DSM) is a problem
- All facets of design are getting more complex (continuously)
- Therefore, we need to make continuous (means incremental?) improvement to tools/design methods.

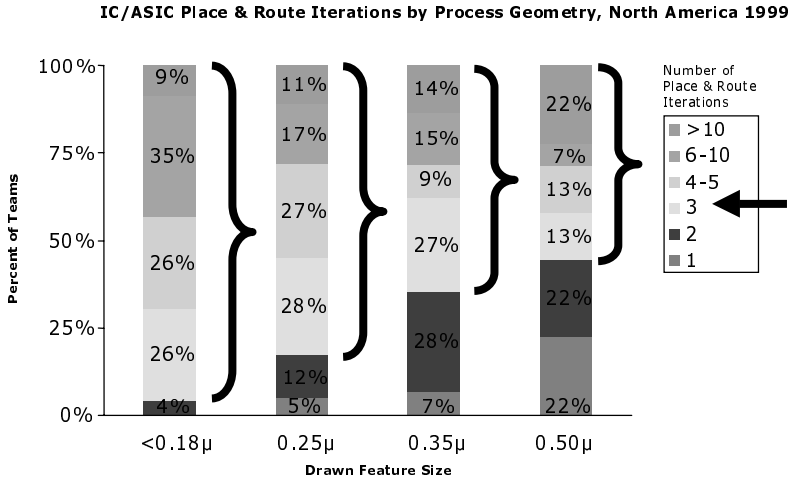
SOC / DSM Design Dilemma



Require detailed analyses to understand physical interactions

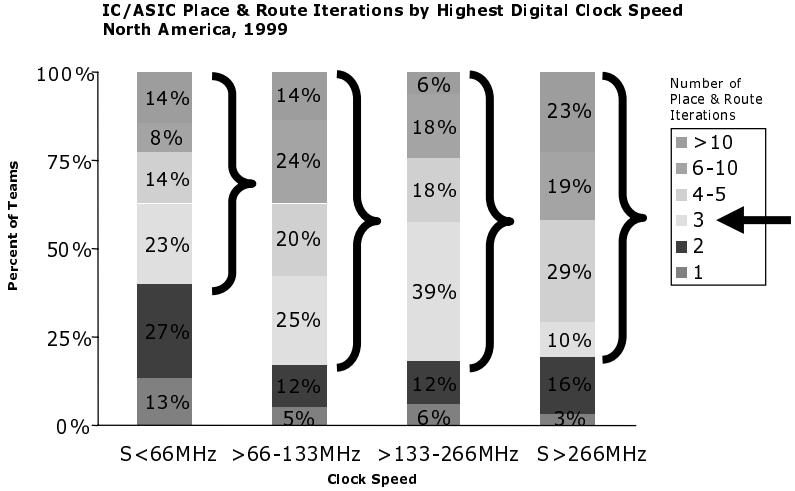
Need abstraction levels to manage complexity

Feature Size and Iterations



Source: Collett Intl. 1999 IC/ASIC Physical Design & Layout Verification Study.
Data based on 224 North American IC/ASIC product development teams.

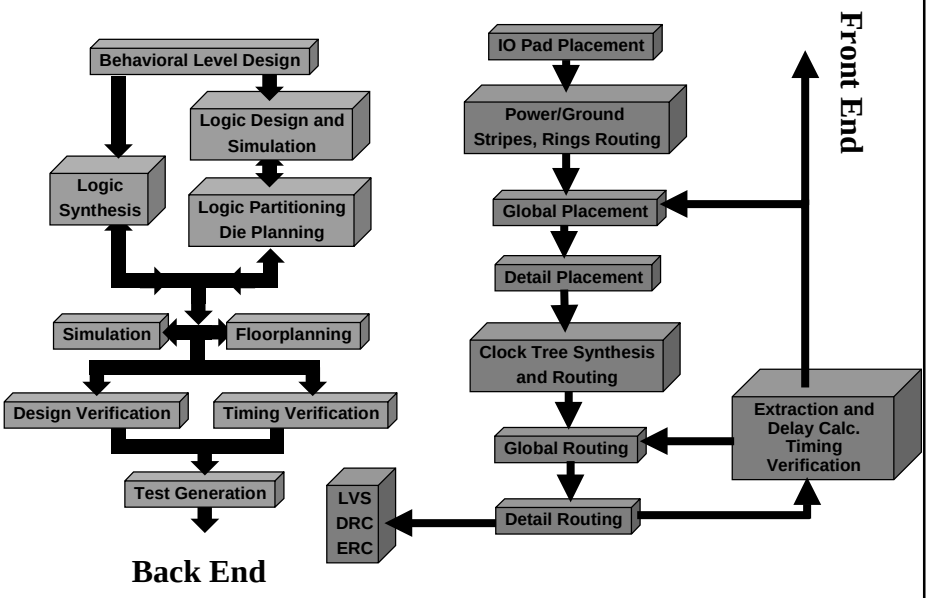
Clock Speeds and Iterations



Source: Collett Intl. 1999 IC/ASIC Physical Design & Layout Verification Study.
Data based on 220 North American IC/ASIC product development teams.

Front End Flow

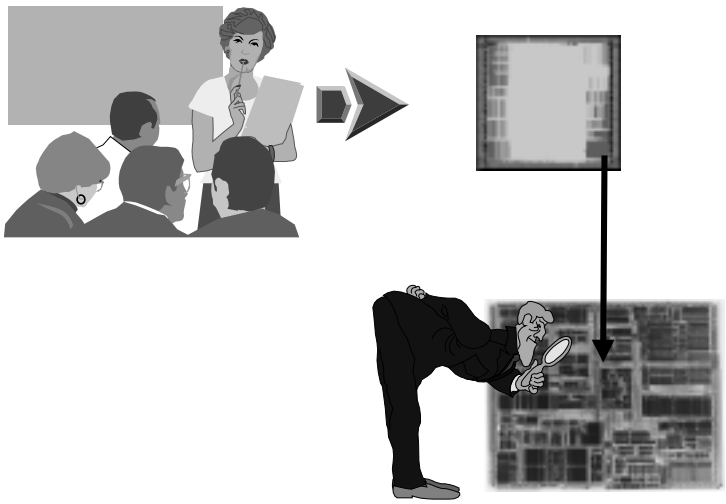
Back End Flow



Observation 2

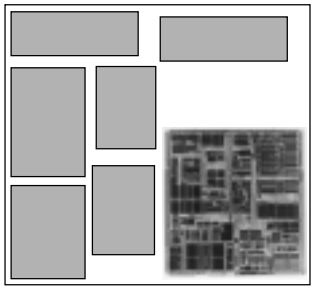
- Therefore, we need to make continuous (means incremental?) improvement to tools/design methods.
- As designs get more complex, number of iteration increases rapidly.
- Incremental (or no) improvement to tools will work (if we are willing to wait 365 days for the result)…. And then the marketing tells us that we need a new feature.

An Easy Solution: Re-use cores



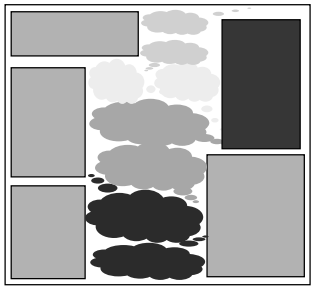
Hard IP

- The easiest path to SoC (?)
- Hard blocks makes the assembly more difficult
  - see results in the next two slides
- No resizing capability to fix timing during assembly



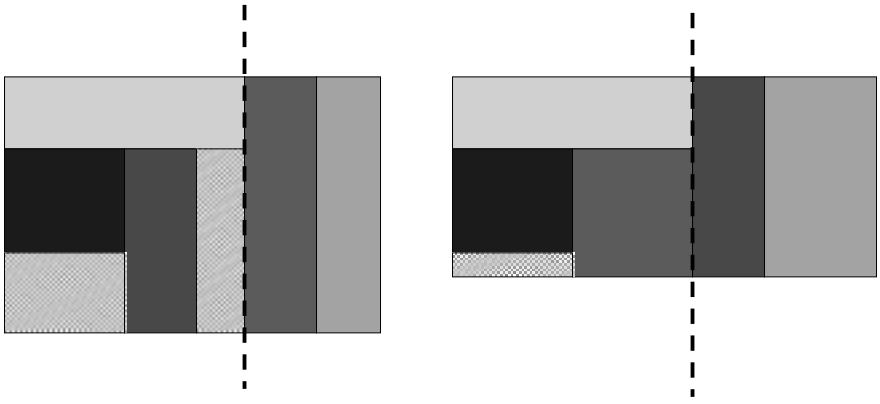
Soft IP

- "Soft" IP will allow better Global Optimization
- Final assembly may solve shape, pin, and global timing problems causing reduced design iterations.



Prediction/Construction heuristics

- Balance with respect to area and flexibility (if there are a lot of flexible/soft IP).



Results

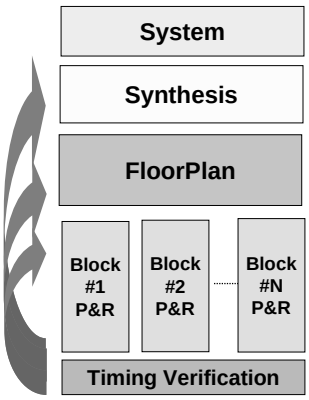
ckt(%rigid)	Wong-Liu		Constructor			
	cost	time(sec)	cost	% diff	time(s)	speedup
ind1(10)	124939	177	121852	-1.18	8.6	21
ind1(20)	121681	173	123241	1.28	7.5	23
ind1(30)	124971	122	125480	2.86	5.9	21
ind1(50)	116033	88	117578	9.64	5.1	17
hway(10)	70571	3420	68611	-1.36	184	18
hway(20)	71838	3515	70448	-1.93	183	19
hway(30)	72274	3524	71900	-0.51	210	17
hway(50)	77633	2563	78698	1.34	105	24
fract(10)	131431	15651	128388	-2.32	897	18
fract(20)	137044	15803	130694	-4.42	704	18
fract(30)	137084	14694	135869	-0.88	723	20
fract(50)	144972	9268	145592	0.91	540	17
prim1(10)	831329	110491	832365	1.2	4629	24
prim1(20)	867690	100010	862657	-0.6	4011	20
prim1(30)	870456	95299	871623	0.13	4214	23
prim1(50)	897120	60303	931694	5.65	3617	19
prim2_s1(10)	230703	11899	215193	-6.72	455	26
prim2_s1(20)	235604	11141	217542	-5.20	430	26
prim2_s1(30)	248317	9306	229349	-7.46	411	23
prim2_s1(50)	249489	7445	248566	-0.37	328	23
prim2_s2(10)	323017	38416	283188	-12.33	1477	26
prim2_s2(20)	319097	30062	301768	-5.67	1309	22
prim2_s2(30)	333313	29632	308023	-7.59	1506	20
prim2_s2(50)	354387	21045	323188	-8.80	1138	18

Divide and Conquer

- Divide the Problem into Smaller Sub-Problems
- Solve Each of these Separately
- Stitch together the Solutions of the Sub-Problems

10 Million Gate Design => 200 (50k Gate Designs)

Divide and Conquer



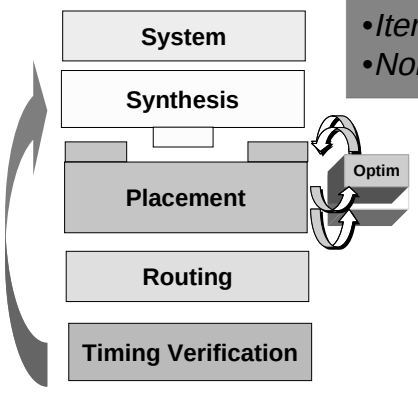
- Divide into Logical/Physical Blocks
  - Particularly emphasize the floorplan
  - Iterations between different tools
- Traditional floorplan
  - No flexibility to fix timing problems caused by long wires
  - Overly constrained timing budgets
  - Adds many buffers and oversizes gates on critical paths

- *Predictive (*we lack good predictors*)*
- *Iterative (*takes a long time*)*
- *Non-converging (*counter productive*)*

Sequential Methodology

- Try to Solve the Problem in Sequential Steps
- Try to Optimize One Functionality at a Time.
  - Optimize Number of Gates at the Logic Synthesis Level
  - Optimize Wire Lengths during Placement
  - Optimize Clock Skew After Placement is Done
  - Optimize Crosstalk during Routing

Sequential Methodology



- *Predictive (*we lack good predictors*)*
- *Iterative (*takes a long time*)*
- *Non-converging (*counter productive*)*

Nets that meet timing in one iteration may fail in the next iteration

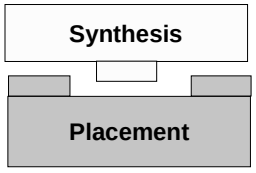
Observation

- There are many “equally good” placement and routing solution. A small change in one, will change the whole things.
- So, cannot trust wire-load models

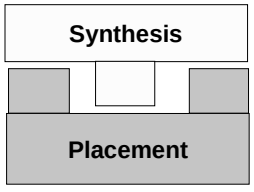
Traditional Workarounds

- Pessimistic approach
  - For 50K block size, use wire-load model for 100K instead
  - Nets are over-driven
  - Wastes power and area, but reduces number of nets that need fixing after phys design
  - Assumes timing can be met with the pessimistic model (not always the case)
- Over-constrained approach
  - For 80 MHz design, synthesize at 100 MHz
  - After physical design, reset to 80 MHz
  - Nets between 80-100 MHz will “pass”
- Multiple-iteration approach
  - Annotate timing info and phys design info into P&R and synthesis
  - Optimization attempts to minimize changes to accuracy of phys design (usually can’t do)

Semi-Sequential Methodology

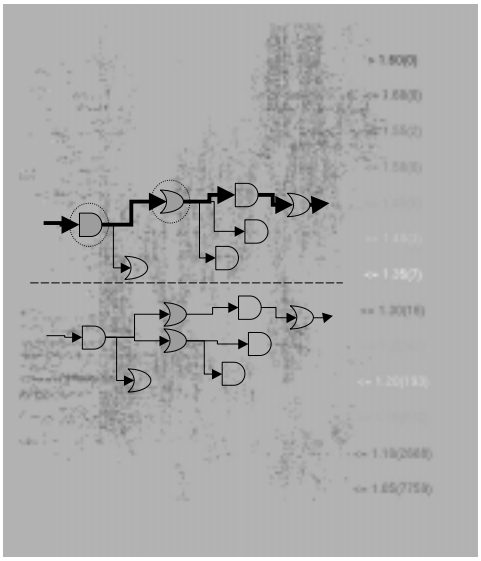


•Lots of logic move followed by lots of placement move

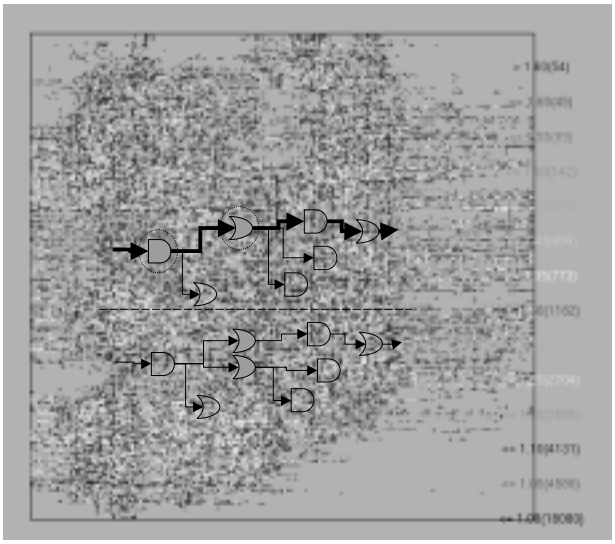


•Some logic moves followed by some placement moves

Low Congestion: some logic activities to CORRECT synthesis mistakes

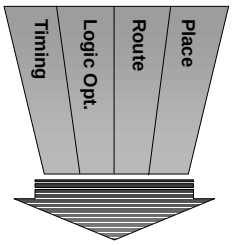


High Congestion: lots of logic activities (panic mode)



Simultaneous Methodology

- "combine" placement and synthesis (& other steps)
- We need to find the right type and location of the move.



Proof of Simultaneous Methodology

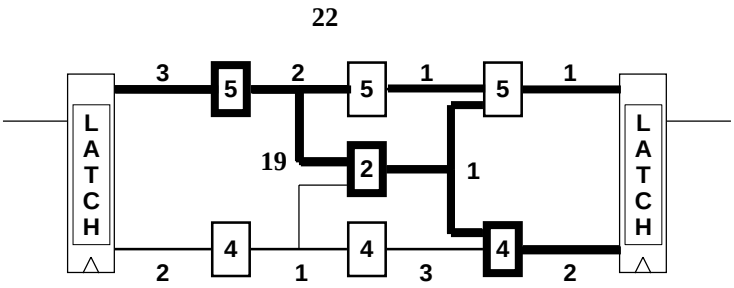
- Obviously, the most knowledgeable set of moves
- haven't been done in the past because
  - history
  - algorithmic complexity
  - need

Timing Optimization

Gate Delays as well as Interconnect delays needs to be an essential part of the design process.

Static Timing Analysis needs to be integrated into the optimization process.

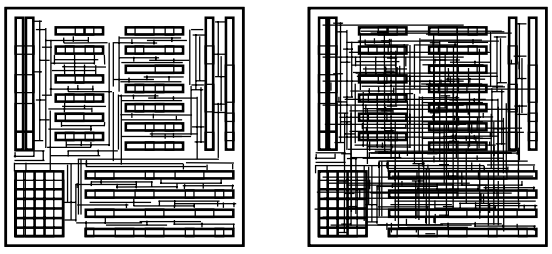
Timing Analysis



How do we get the delay numbers on the gate/interconnect?

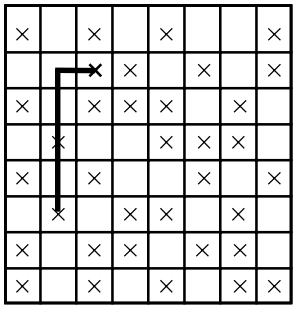
Timing Metrics

- How do we assess the change in a delay due to a potential move during physical design?
- Whether it is channel routing or area routing, the problem is the same
  - translate geometrical change into delay change



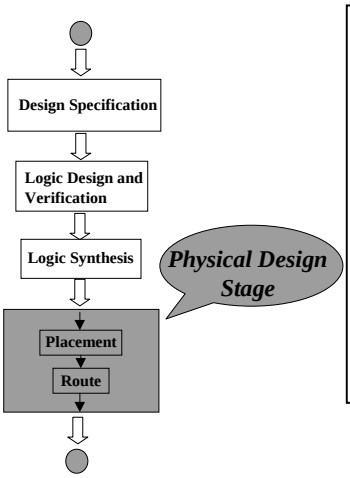
Iterative Placement

- A placement move changes the interconnect capacitance and resistance of the associated net
- A net topology approximation is required to estimate these changes



Placement Algorithms

VLSI Design Flow and Physical Design Stage

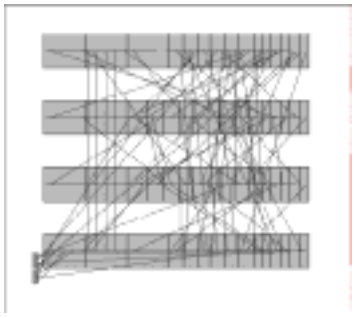


A flow chart for a typical VLSI design.

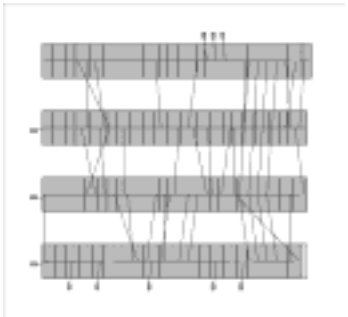
Definitions:

- *Cell: a circuit component to be placed on the chip area. In placement, the functionality of the component is ignored.*
- *Net: an electronic wire to connect several cells.*
- *Netlist: a set of nets which contains the connectivity information of the circuit.*

Placement Problem

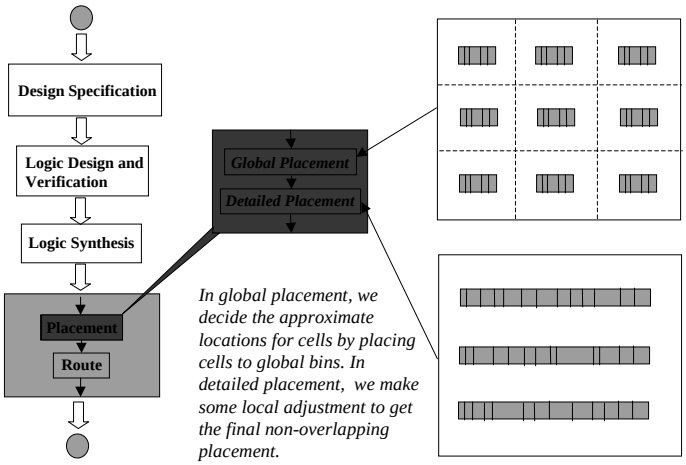


A bad placement



A good placement

Global and Detailed Placement

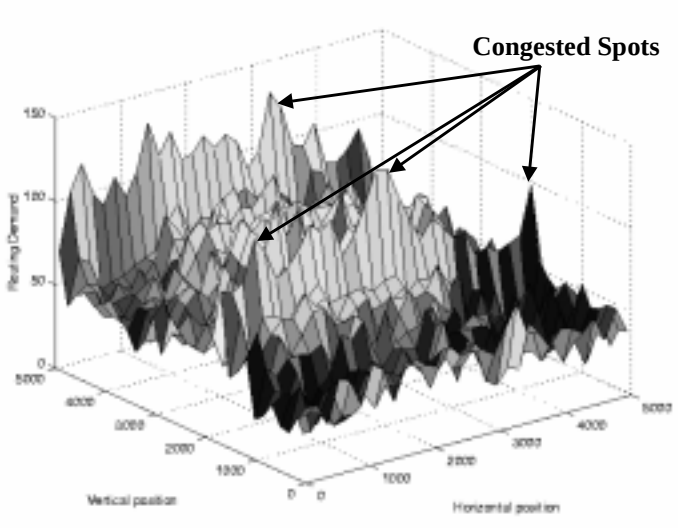


In global placement, we decide the approximate locations for cells by placing cells to global bins. In detailed placement, we make some local adjustment to get the final non-overlapping placement.

Traditional Approaches

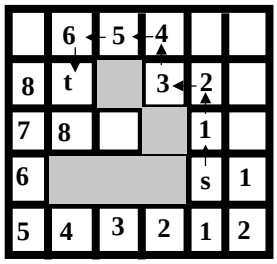
- Quadratic Placement
- Simulated Annealing
- Bi-Partitioning
- Quadrisection
- Force Directed Placement
- Hybrid

Congestion Map for a Wirelength Optimized Placement



Routing Algorithms

Maze Routing/Shortest Path

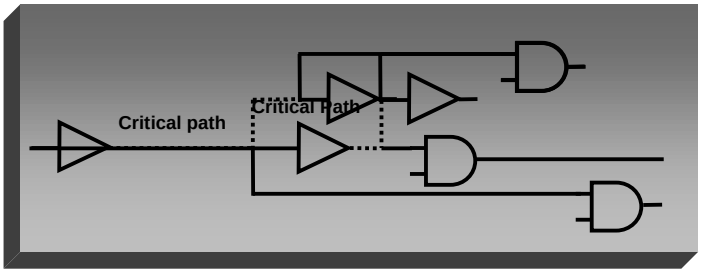


Label Source s = 0
Label Adjacent grid points using *i*+1
Get Shortest path to target t

Routing

- Requirements for the DSM Router:
  - N-layer shape-based router
  - Supports gridless and gridded routing
  - Variable wire width for optimal delay constraints
  - Cross-talk avoidance, antenna effects
  - Clock tree sizing for tree balancing
  - Power routing sizing for voltage drop and electromigration
  - Power and clock routing resources reserved early
  - Activity-based optimization

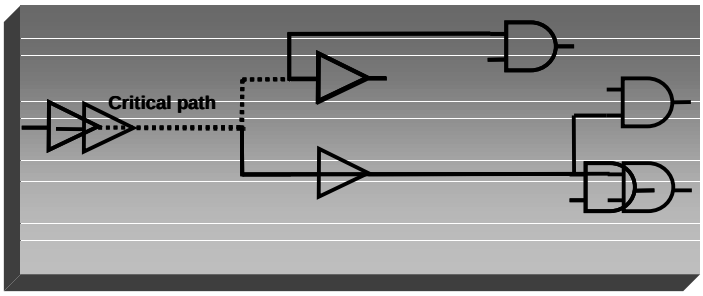
Typical Timing-Driven Approaches



Timing Driven Placement

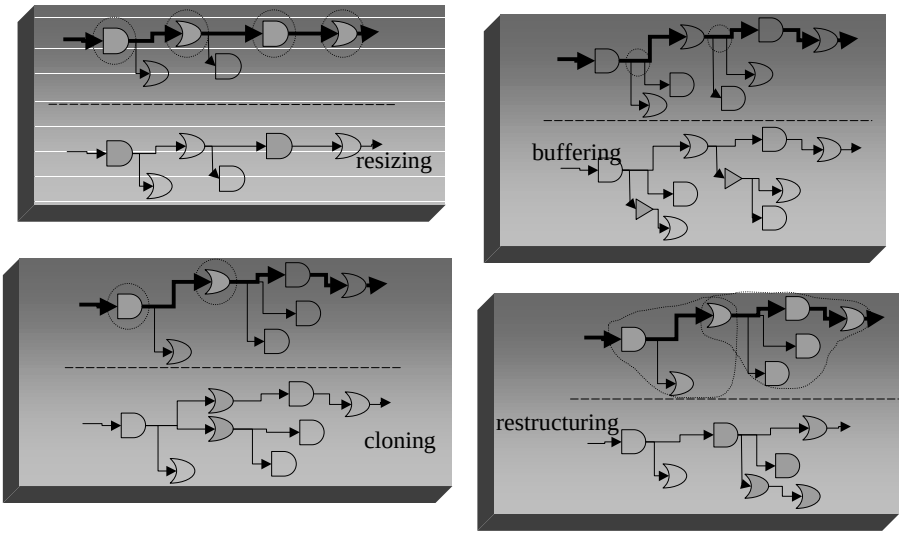
- **Net weighting to prioritize timing critical Paths**
- **Reduction of entire net length**

Placynthesis: Simultaneous Logic/Placement Approach

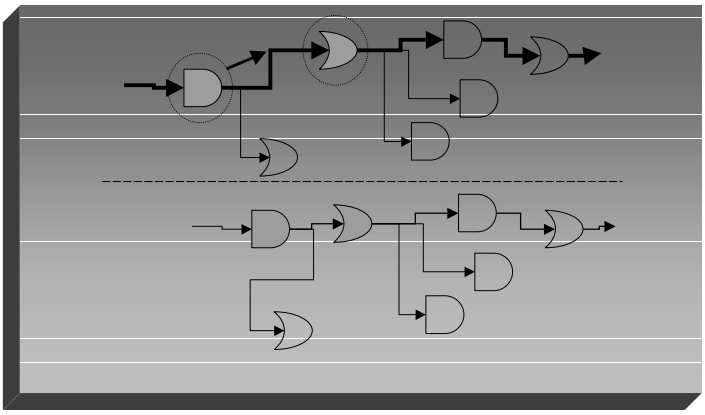


- **Logic optimization concurrently with placement**
- **Net "placement" with gate placement**

Some Placynthesis Moves

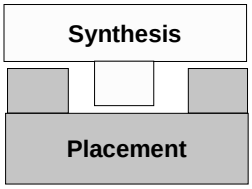


More Placynthesis Moves

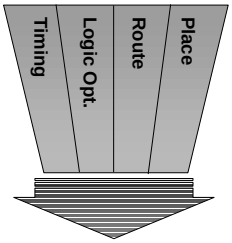


Commercial Integration Approach:

- Integrate synthesis with phys design
  - Cadence (Envisia Synthesis, 9/1999)
    - Physically Knowledgeable Synthesis (PKS)
  - Synopsys ("PhysOpt")
  - Monterey ("Doplhin")
  - Magma ("??").

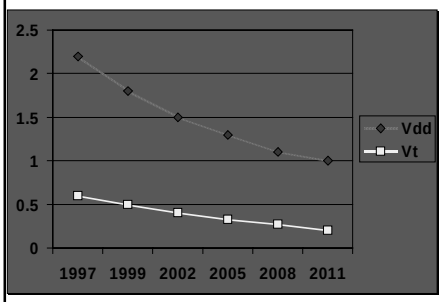


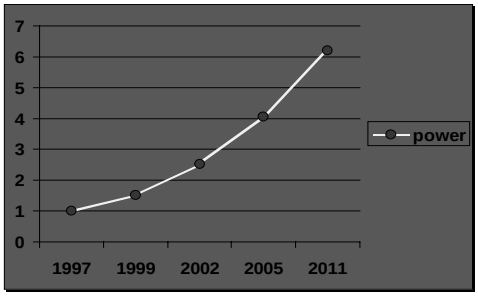
Semi-Sequential Methodology



Simultaneous Approach

Many other Design Metrics: Power Supply and Total Power

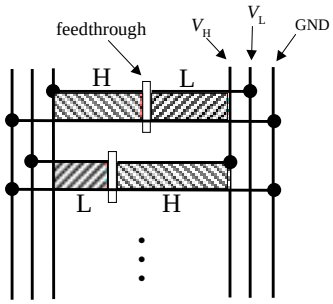




Source: The Incredible Shrinking Transistor, Yuan Taur, T. J. Watson Research Center, IBM, IEEE Spectrum, July 1999

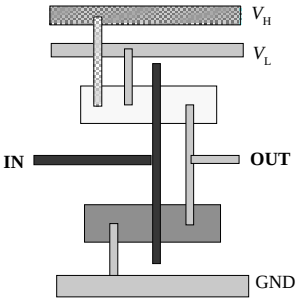
Dual Voltages: A harder problem

- Layout synthesis with dual voltages: major geometric constraints



H -- High Voltage Block
L -- Low Voltage Block

Layout Structure



Cell Library with Dual Power Rails

Conclusion

- Deep Sub-micron (DSM) problems are here and are real
- Traditional Physical Design and Logic Synthesis Algorithms do not work
- Innovation (in algorithms, methodology, tools, etc) needed in all facets.