Efficient Timing Closure Without Timing Driven Placement and Routing

Miodrag Vujkovic, David Wadkins, Bill Swartz* and Carl Sechen

Dept. of Electrical Engineering
University of Washington
Seattle

*InternetCAD.com
Dallas, TX

Outline

- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

Prior Work

1. C.K. Cheng, “Timing Closure Using Layout Based Design Process”, (www.techonline.com/community/tech_topic/timing_closure/116).
2. R. Bryant, *et al*., “Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI”, *Proc. IEEE,* Vol. 89, No.3, March 2001.
3. O. Coudert, “Timing and Design Closure in Physical Design Flows”, *Proc. of the International Symposium on Quality Electronic Design (ISQED)*, 2002.
4. S. Hojat, and P. Villarrubia, “An Integrated Placement and Synthesis Approach for Timing Closure of PowerPC TM Microprocessors”, *Proc. IEEE Int. Conference on Computer Design (ICCD),* pp. 206-210, October 1997.
5. B. Halpin, N. Sehgal, and C.Y. R. Chen, “Detailed Placement with Net Length Constraints”, *Proc. of the 3rd IEEE Int. Work. on SOC for Real-Time Applications*, 2003.
6. S. Posluszny *et al*., “Timing Closure by Design, A High Frequency Microprocessor Design Methodology”, *Proc. of Design Automation Conf. (DAC)*, pp. 712-717, June 2000.
7. G. Northrop, and P.F. Lu, “A Semi-Custom Design Flow in High-Performance Microprocessor Design”, *Proc. of Design Automation Conference (DAC)*, pp. 426-431, June 2001.
8. E. Yoneno, and P. Hurat, “Power and Performance Optimization of Cell-Based Designs with Intelligent Transistor Sizing and Cell Creation”, *IEEE/DATC Electronic Design Processes Workshop*, Monterey, CA, April 2001.
9. M. Hashimoto, and H. Onodera, “Post-Layout Transistor Sizing for Power Reduction in Cell-Base Design”, *IEICE Trans. Fund.,* Vol.E84-A, pp. 2769-2777, November 2001.

Problem Domain

- The focus of this paper is timing closure at the block level
  - *e.g.* up to 50k cells -- or library instances
  - but certainly scaling up as tools (gate sizers) improve
- The approach certainly can be applied hierarchically, to obtain similar benefits for much larger circuits

The Challenge!

- What makes the timing closure problem particularly difficult is the high variability in the loading of wires
- We measured the capacitance per unit length for all nets for a wide variety of circuits in the 0.18um TSMC technology
  - varied from about 0.02 fF/um to about 0.7 fF/um
  - a 35X variance!
- The value you get for a particular net is only ascertained AFTER detail routing
- There is no way for a timing driven placement approach to be effective in meeting timing
  - Predicting useful gate loads seems impossible

Design Flow Objectives

1. Absolutely minimize energy (power) for a specific delay target
2. Rapid timing convergence

Concepts Behind the Objectives

1. Absolutely minimize energy (power) for a specific delay target
   - Use only combinational gates that are power efficient
   - Provide a very wide range of drive strengths and beta ratios for this limited set of gates
   - Transistor area only used if absolutely necessary to meet delay constraint

2. Rapid timing convergence
   - Since timing analysis must be done while gate sizing (actually, gate selection) anyway, ONLY do TA there!
   - Since wire loads are crucial to power & delay, want placer to only worry about wire lengths
   - Placer objective function is not concerned with: congestion, timing, overlap, *etc*., rather, only wire length
   - Enables MUCH lower wire lengths, hence power & delay
   - Also need effective ECO mode, featuring “stable” placers and routers (about the same wire lengths after an ECO iteration)

Outline

- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

Cell Library

- A cell library consisting of 14 different combinational functions appears to be optimal for minimizing power for a specific delay (ICCAD 2002)
  - INV, NAND2-4, NOR2-3, AOI/OAI12, AOI/OAI22, XOR, MUX2:1
  - FA (sum), FA (carry)
- Pass-transistor versions of XOR and FA also available
  - pass transistor cells yield higher speed, but are higher power
- Crucial to provide WIDE range (and number) of drive strengths
- Even more crucial to provide very wide range (and number) of beta ratios

Cell Sizes

- All channel-connected nMOS devices for a cell are sized as a group
- All channel-connected pMOS devices are sized as a separate group
- Series/parallel cells have only one group of nMOS devices and one group of pMOS devices while pass-transistor cells can have several nMOS/pMOS transistor groups
- For larger widths, the sizes correspond to an integral number of half folds (*i.e*. transistor widths are 1.0, 1.5, 2.0, 2.5, *etc*., times the maximum non-folded size)
- For a 0.18um process, the allowable sizes for both nMOS and pMOS devices range from 1.06um to 13.78um, in steps of 0.53um
  - From 0.42um to 1.06um, approximately in steps of 0.13um
- Total number of instances of 14 combinational functions: about 1300

Cell Library (cont'd)

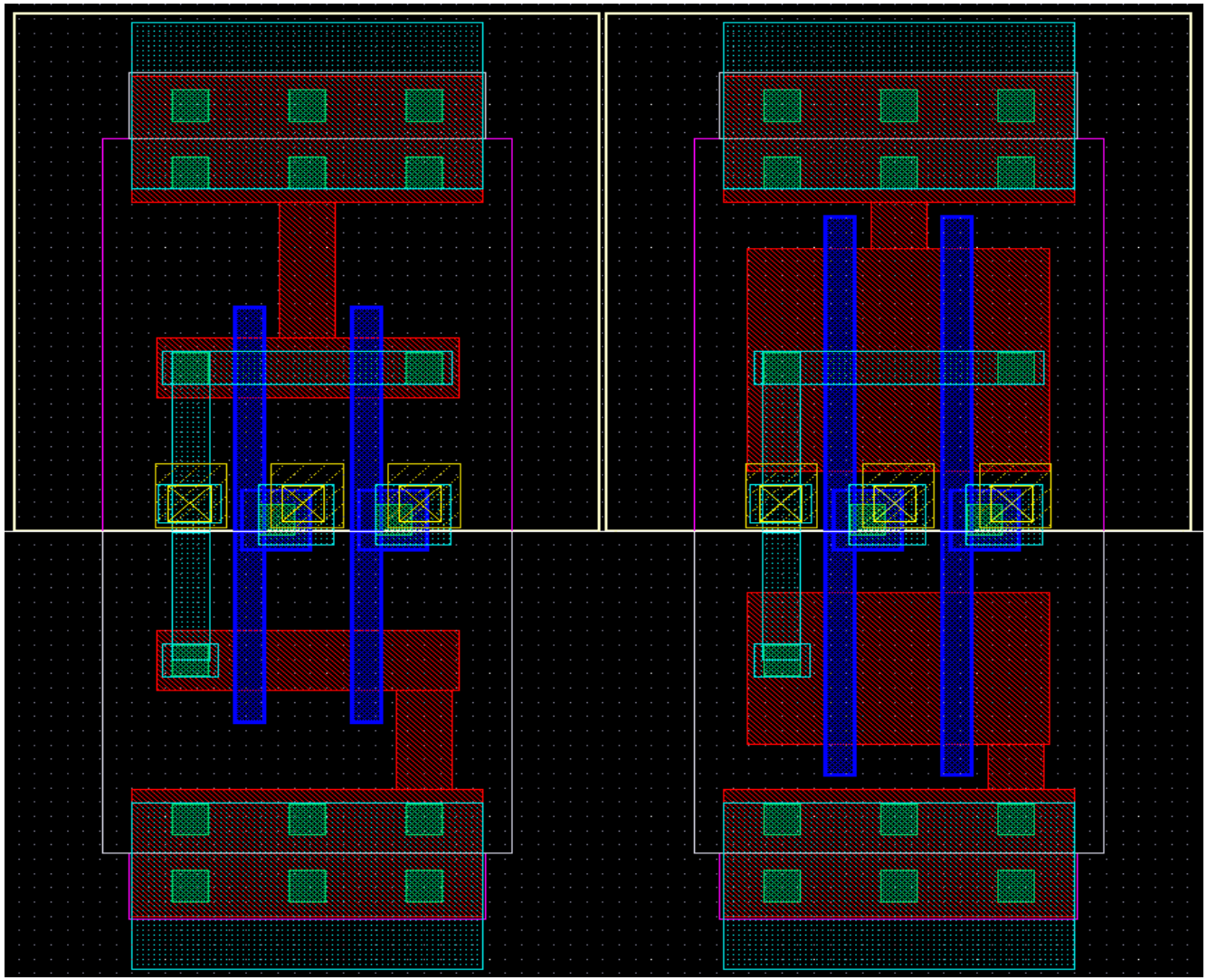
- Fixed library approach!
  - library is characterized and validated via fabrication
- During synthesis (only), the library is enhanced to include various cells that are "compounds" of the 14 base functions
  - greatly enhances flexibility during synthesis without using power inefficient gates, and without enlarging the physical library

Parameterized Cell Layout Generator

- Foundry-independent, parameterized, automatic cell layout generator developed
- Separate generator for each of the 14 combinational cell types plus DFF cells
- Cell layouts specially tuned to yield best possible placement and routing density

NAND2 Gate

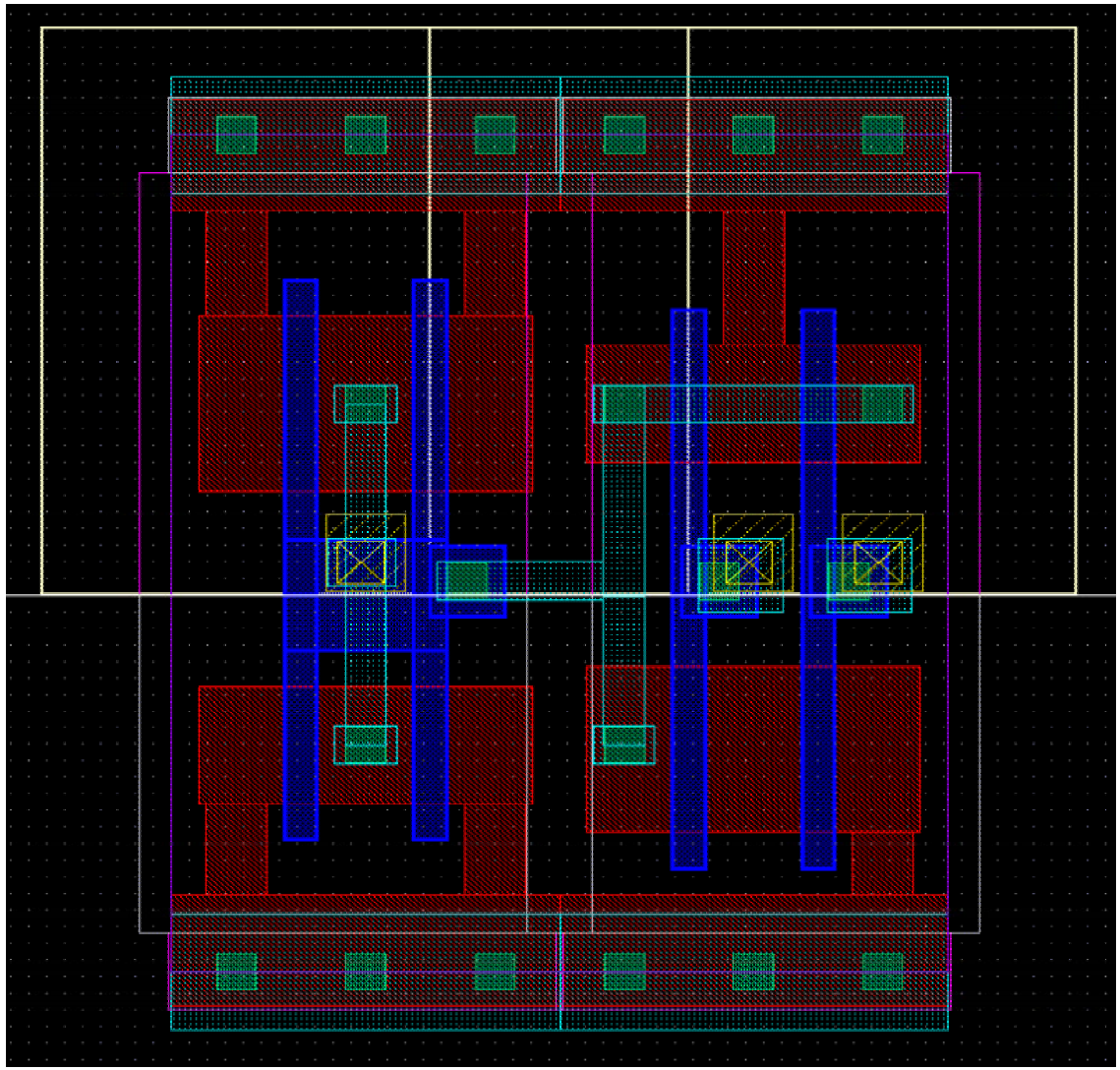
- nMOS size: 0.42u to 1.06u within one fold
- pMOS: from 0.42 u to 1.56u within one fold



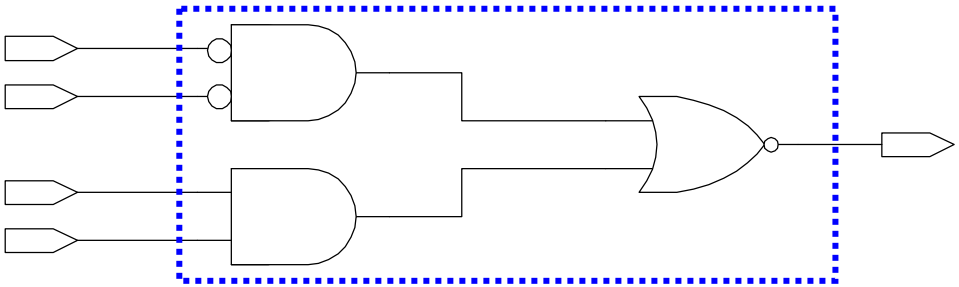
Compounding Cells

- Gates (base functions) comprising a compound cell are sized separately
  - Improves power efficiency
- However, great benefit in HARDWIRING these gates together:
  - One circuit: # of nets to route decreased from 8000 to 5000 from hardwiring the compound cells
  - 2nd circuit: # of nets to route decreased from 19,000 to 13,500 from hardwiring the compound cells
  - Not surprisingly, in each case the routing area decreased by 60%!

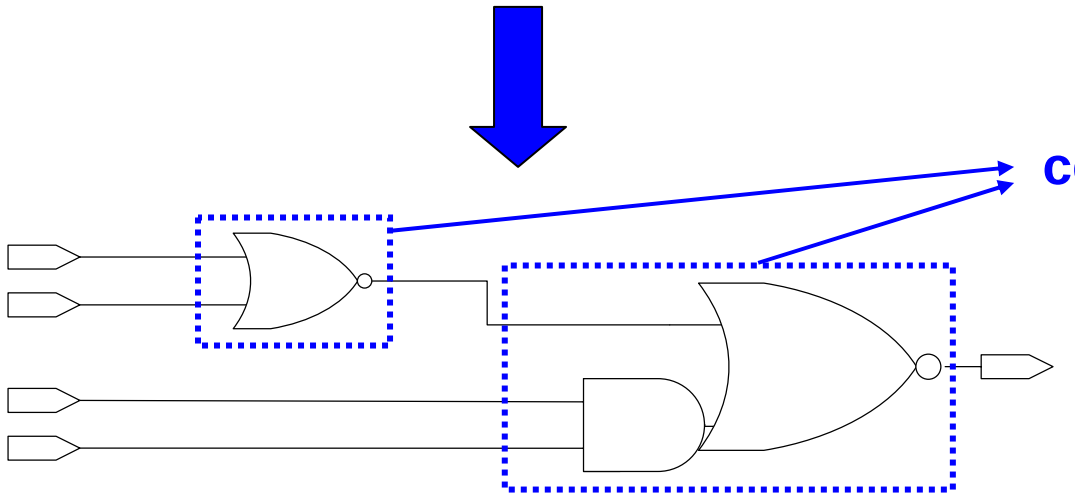
Compound Cell Example: AND2



Example of Compound Cell

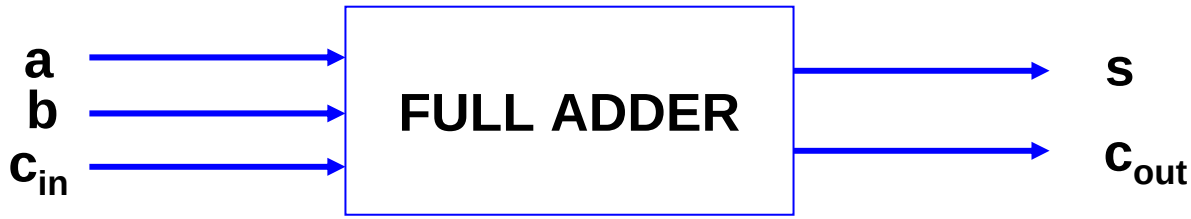


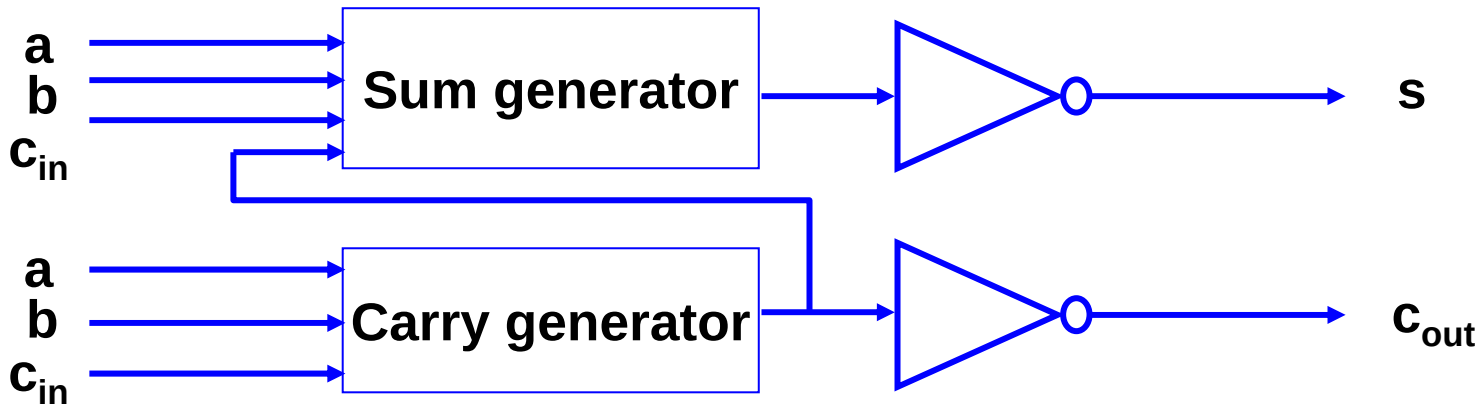
AOI2BB2 cell



cells separately optimized

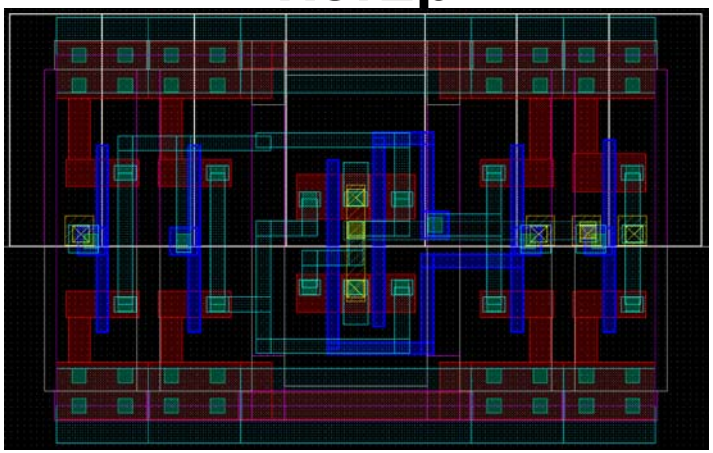
Another Compound Cell Example



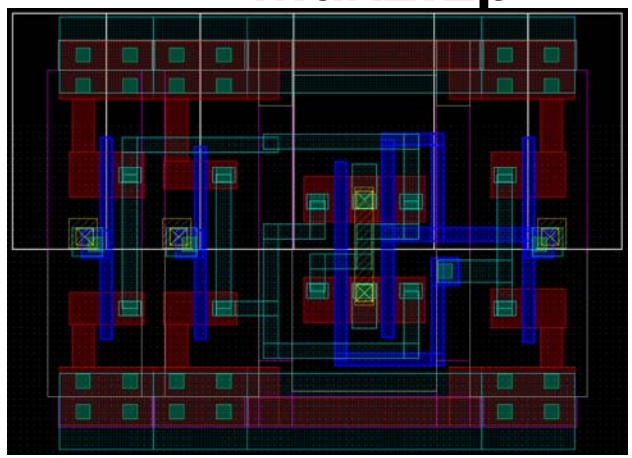


Cellgen Cells

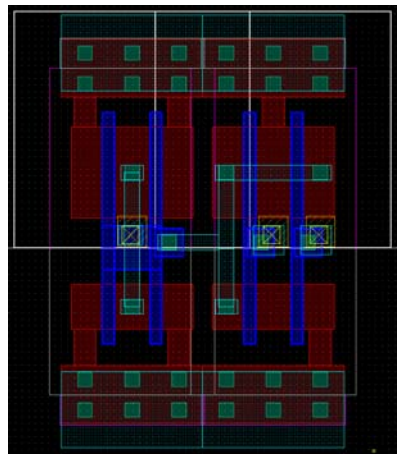
xor2p



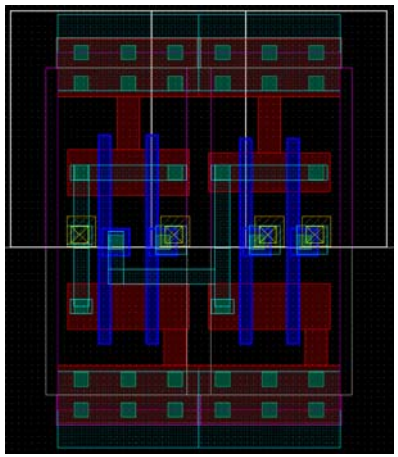
mux2:1p



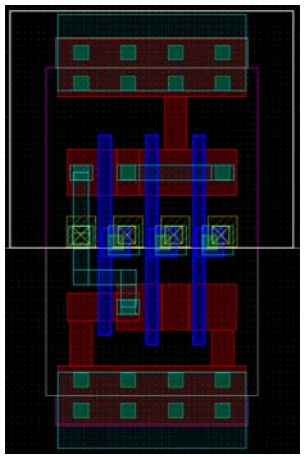
and2



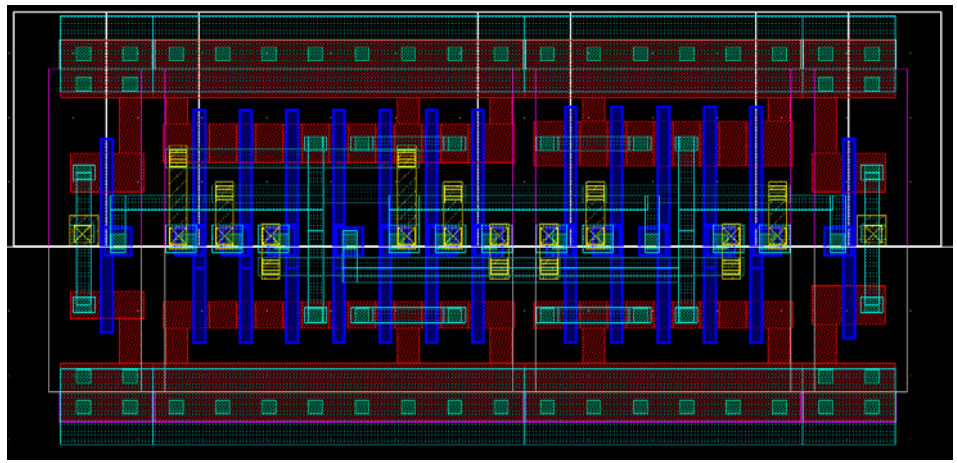
oai2bb1



aoi21



addfs



Outline

- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

Placement Issues

- All timing analysis resides within the cell sizer, where the cells have their sizes optimized for the actual extracted wire loads
- A critical advantage of this new approach is that the placement tool can focus on only a single objective, that of minimizing wire length
  - But we don’t allow any single net from becoming “too long” … by providing an upper bound
- We need an approach that generates placements having the lowest possible total wire lengths

Placement Issues (cont'd)

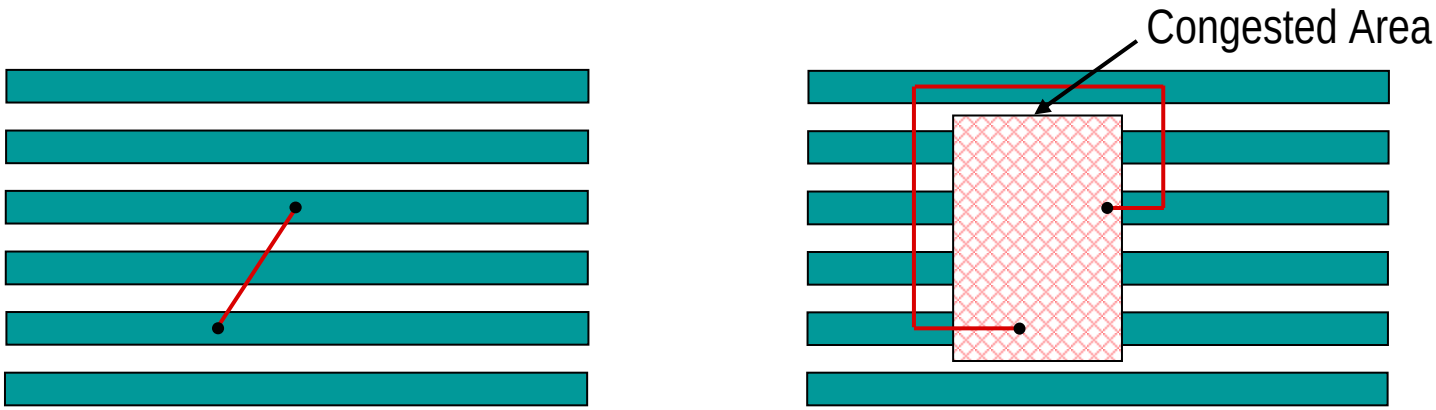
- Based on our experience, the iTools placer from InternetCAD.com yields lower total wire lengths compared to other EDA placers
  - For the smaller PEKO benchmarks (having known optimal wire lengths), iTools yields results around 10% above optimal
  - Other placers are in the 40% above or more range
- Run time is not an issue
  - Run time around 3-4 hrs for 20k cells for one PC
  - Parallel mode gives linear speedup with # of PCs or processors
- Also, since resizing cells perturbs the initial global placement, we need the effective ECO (engineering change order) mode in iTools
  - Want about the same wire lengths after cells have been resized
  - Cells are only permitted to move up or down one row, and a limited distance left or right

Routing: Fixed Die vs. Variable Die

- The major EDA companies and much of academic research for more than 10 years have employed what are essentially gate array (or fixed die) routers for standard cell circuits
- Hardly makes sense since the die is programmable on all mask layers in the standard cell style …
- Variable die routers create routing space wherever needed in order to complete the routing
  - variable die routers inherently complete 100% of the routing whereas in fixed die routing this guarantee is absent
- The routing quality obtained from a fixed die router is highly dependent on the quality of the user
  - numerous iterations to eliminate unneeded routing space and/or to add needed routing space

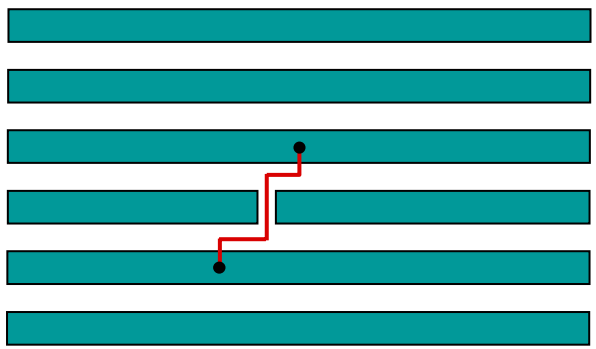
Fixed Die Routers are Bullied by Congestion

- If a congested area lies between 2 pins, the router must detour around it
- Often this net lies on a critical path and now the load for the driving gate is dramatically larger than before, causing timing to diverge
  - Gate sizing creates a lot of critical paths …
- Furthermore, resizing a design often dramatically changes the congestion landscape, causing nets to become much longer (or shorter) than previously (lack of “stability”)
- It is our experience that a fixed die router renders our flow to be largely ineffective in accomplishing timing closure



Variable Die Router

- Each net is routed in near minimum length, every time
- Congestion has no impact
- It simply creates space (*e.g*. increasing row separations or adding feedthroughs) wherever needed to wire up a net using near minimum total wire length
- Thus, a minor placement change, such as what might occur after resizing and an ECO placement run, will yield almost the same lengths for each net
- Great “stability”!

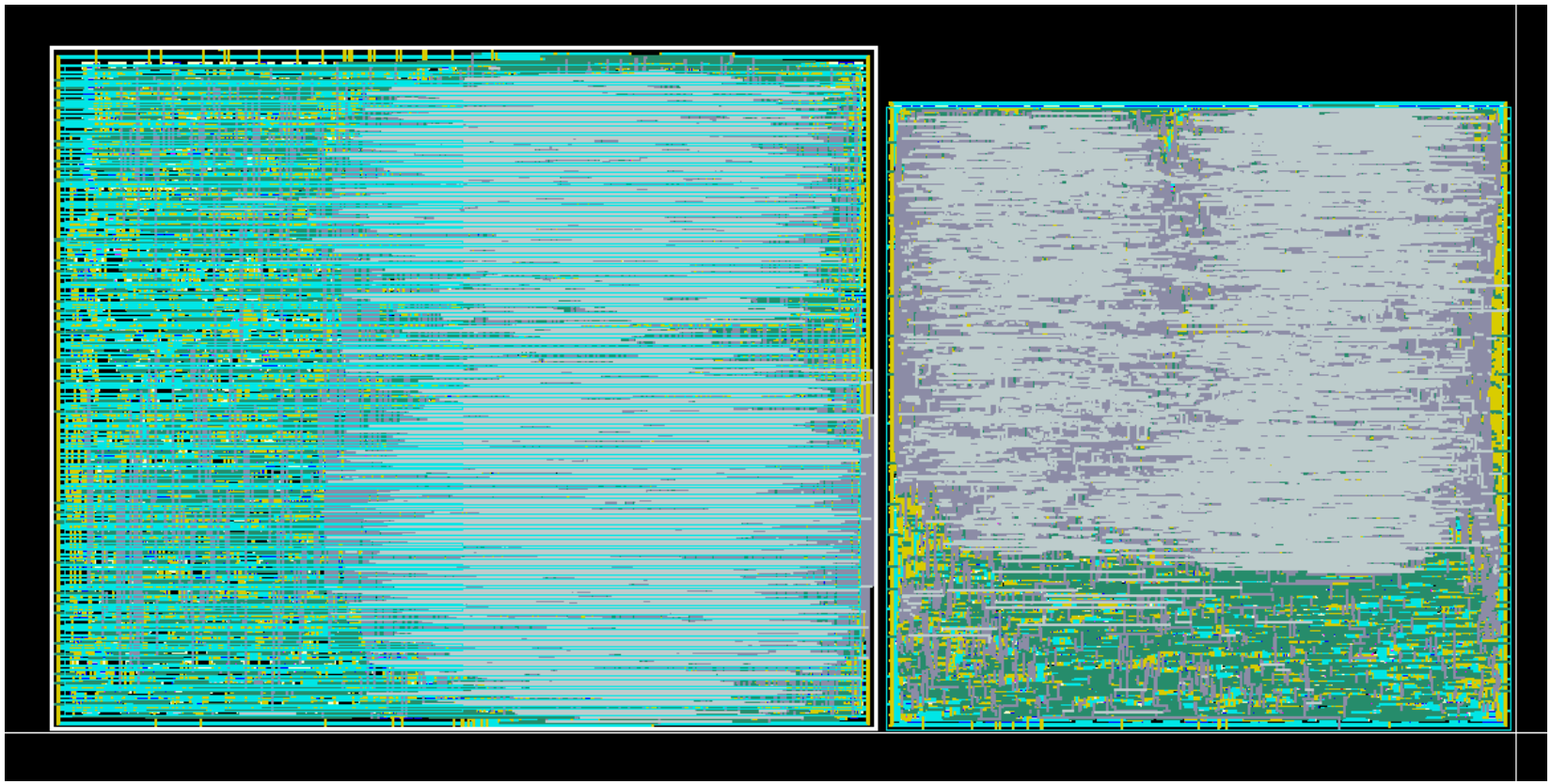


Variable Die Router (cont'd)

- In our flow, we use the iTools gridded variable die router
  - To our knowledge, this is the only commercial (or academic) variable die router available
  - While a non-gridded version is available, the gridded version is faster and our cell library is gridded anyway
  - Parallel algorithm enables linear speedup with # of PCs or processors

Leading EDA Router vs. iTools Routing

- For the same netlist (and library of cells) as well as the same placement, the iTools layout is 33% smaller than the leading fixed die router!
- This design (NCO) has about 8,000 library cells



Outline

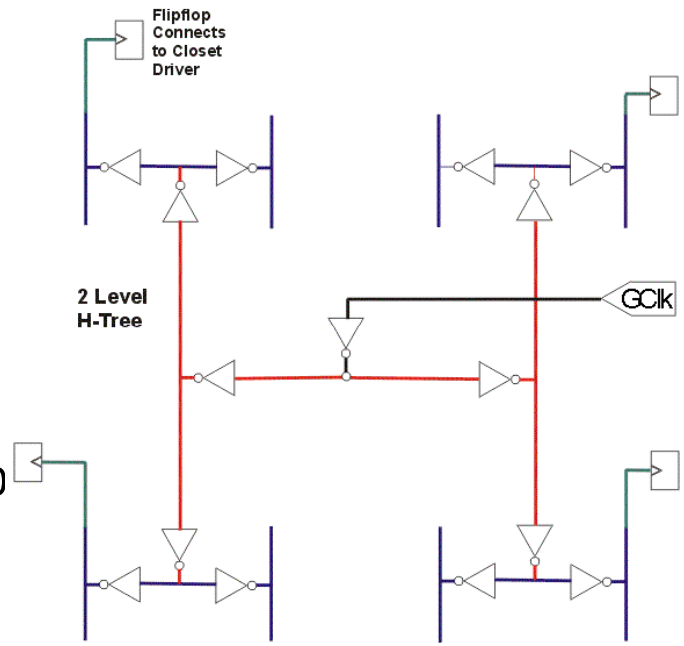
- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

Clock Tree Insertion

- In traditional flows, clock tree synthesis is performed after placement
- This greatly disturbs cell placement, and consequently wire lengths, making timing closure almost impossible

Simultaneous Clock Tree Insertion

- We augmented the iTools placer to add a hierarchical H-tree of symmetrical inverters during the actual placement process
- Just like the rest of the flow, the clock tree synthesis is a refinement process
- Initially, iTools equalizes the transistor loads on the leaf inverters
  - however, it cannot assure an equal RC wire load to each flip-flop during placement since routing is yet to be performed
- After initial global placement, each inverter in the tree is sized using logical effort, based on the capacitive loads



Minimizing and Using Skew

- The result is that the fanout capacitance is constant for each leaf inverter, which in turn tends to equalize the rise and fall time for each inverter and reduces the overall skew
- However, sizing cannot compensate for the skew between flip-flop loads due to RC effects
- The residual clock skews are accurately simulated using Hspice on the Assura (RC) extraction of the clock tree directly to the individual flip-flop loads
- This skew information is incorporated into the gate sizing step, where it is used to optimize the timing

Outline

- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

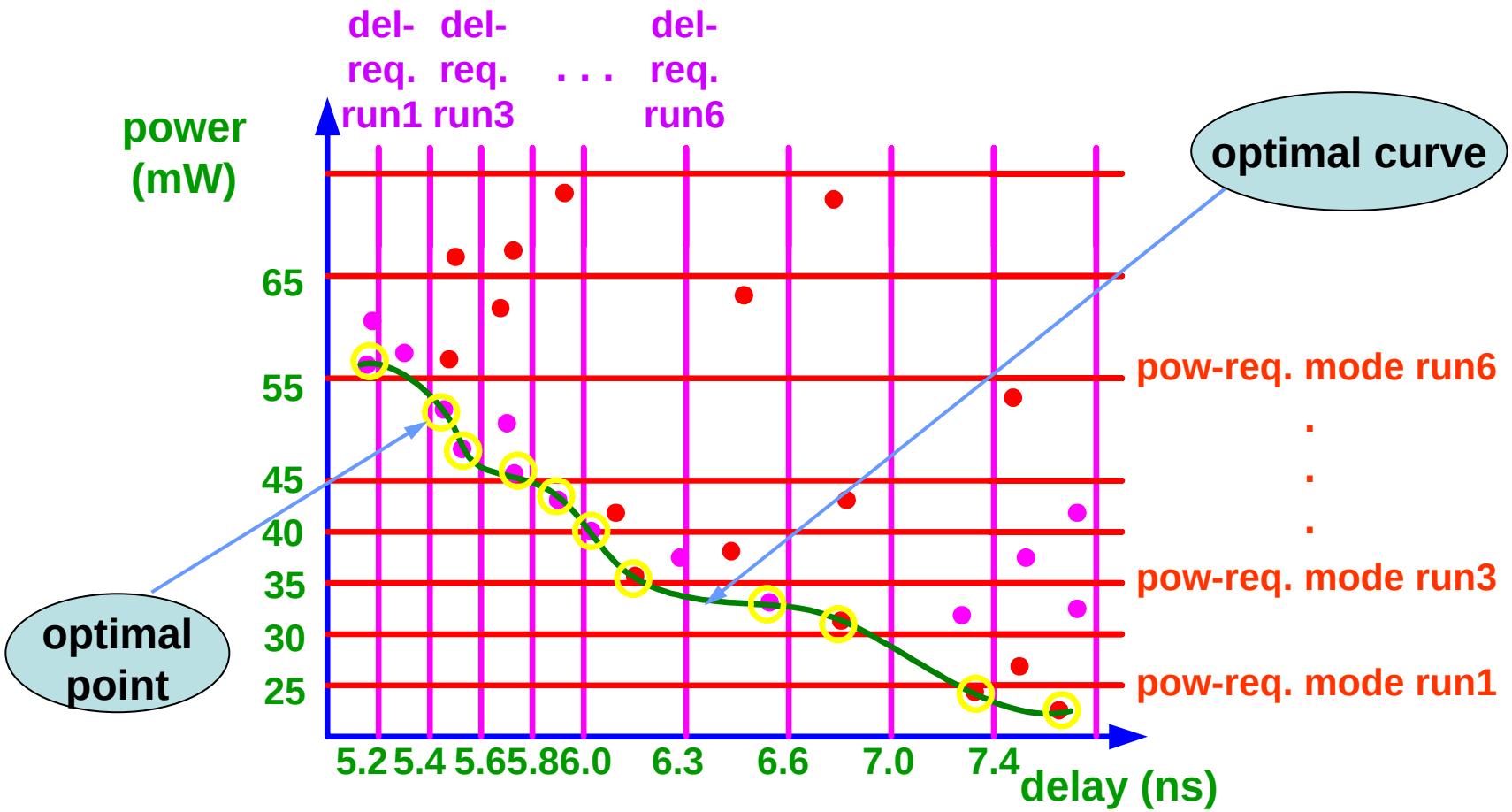
Two Modes of Operation

1. Energy (or power) vs. delay plots
   - Energy is minimized for several delay points or a delay range
   - E vs. D plots can be combined hierarchically, to handle large circuits

2. Minimize energy for specific delay

Generating P vs. D Plots

- Currently use AMPS gate sizer from Synopsys (not really intended for this purpose)
- But, power and delays are fairly accurate
- AMPS is allowed to use only the cell sizes in the library



Outline

- Prior Work
- Problem domain
- Why timing closure is difficult
- Objectives of our design flow
- Cell library
- Variable die placement and routing
- Clock tree insertion
- Energy-delay plots
- Timing convergence design flow details
- Results

Timing Convergence Flow - Overview

1. Synthesis runs using the latest Synopsys products (DC Ultra, BOA, Power Compiler)
2. AMPS runs to determine P vs. D plots (using a wire load model)
3. Select one or more P,D points and execute iTools placement and routing
4. Extract actual wire loads (*e.g*. Assura RCX) and determine P, D
5. Re-run AMPS for one or more of the layout points
6. iTools ECO place and route
7. Extract actual wire loads and determine P, D
8. If necessary, repeat 5-7 once

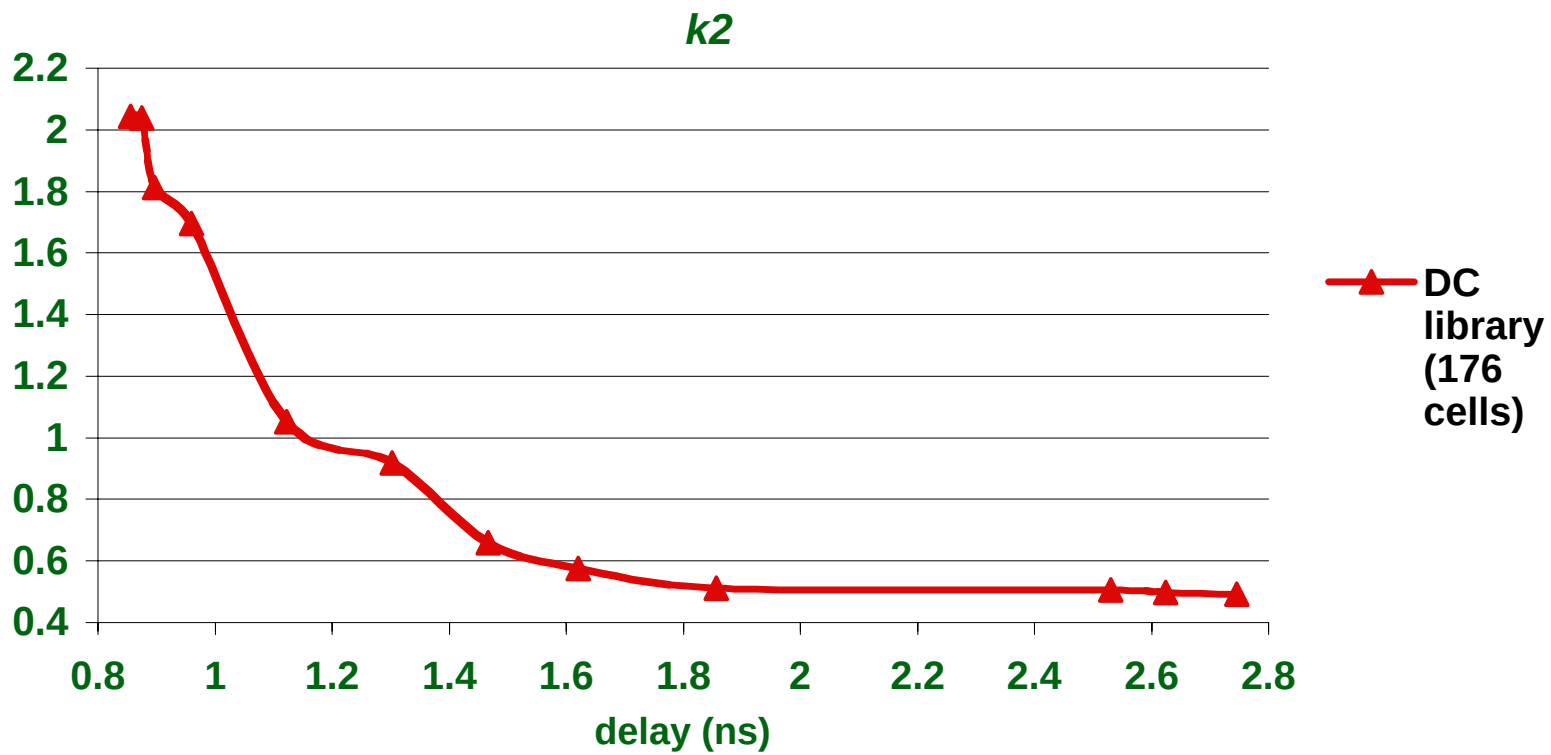
Timing Closure Design Flow - Details

1. Synthesis using Synopsys DC
   - 25-30 iterations or until desired improvement
   - DC-ultra optimization (Behavioral Optimization of Arithmetic)
   - Power optimization (using Power Compiler)
   - Synthesis library includes 156 combinational cells (each decomposes into one or more of the 14 basic functions) and 20 FFs (each decomposes into one of the 5 FFs in our library)

2. Add wire load model to synthesized netlists
   - 17 um/FO (3.4 fF/FO) for TSMC .18um

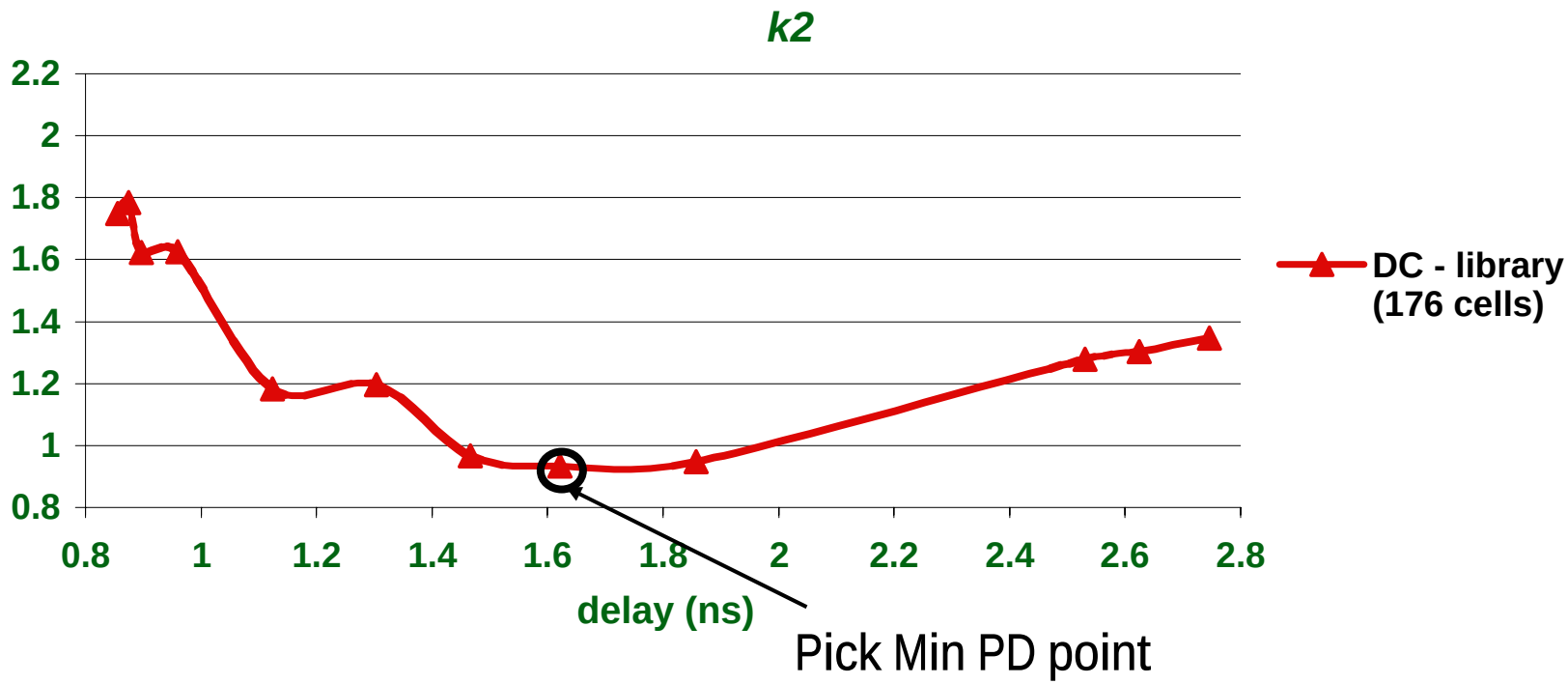
Timing Closure Design Flow (cont'd)

3. Delay/power analysis (Pathmill/NanoSim)
4. "Optimized" curve extraction after synthesis



Timing Closure Design Flow (cont'd)

5. Select point(s) from the "optimized" curve (*e.g.*, target delay, min PD point, min PDA point, or a set of points spanning the P-D curve)
   - Want to run AMPS with the wire load model for certain points

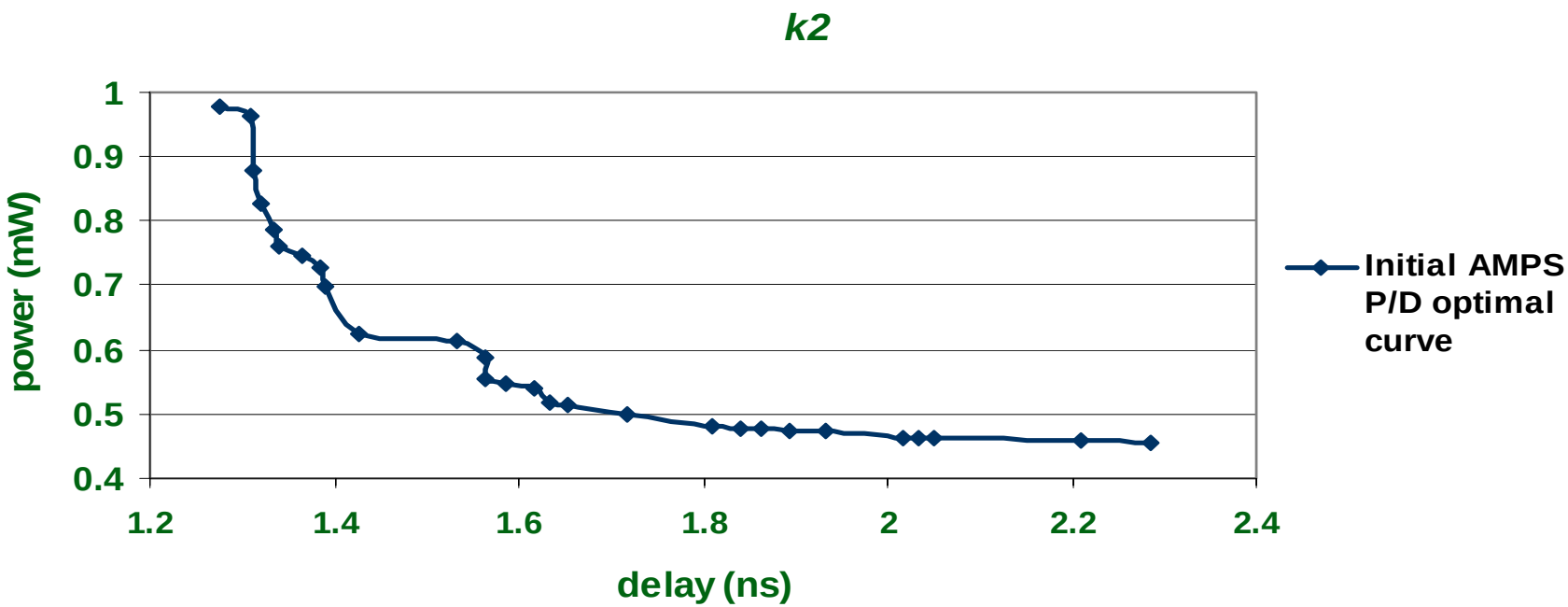


Timing Closure Design Flow (cont'd)

6. Convert compound cells to the set of 14 base functions in our library
   - Full adders are pass transistor for high speed portion of P-D curve; otherwise, static CMOS versions are used

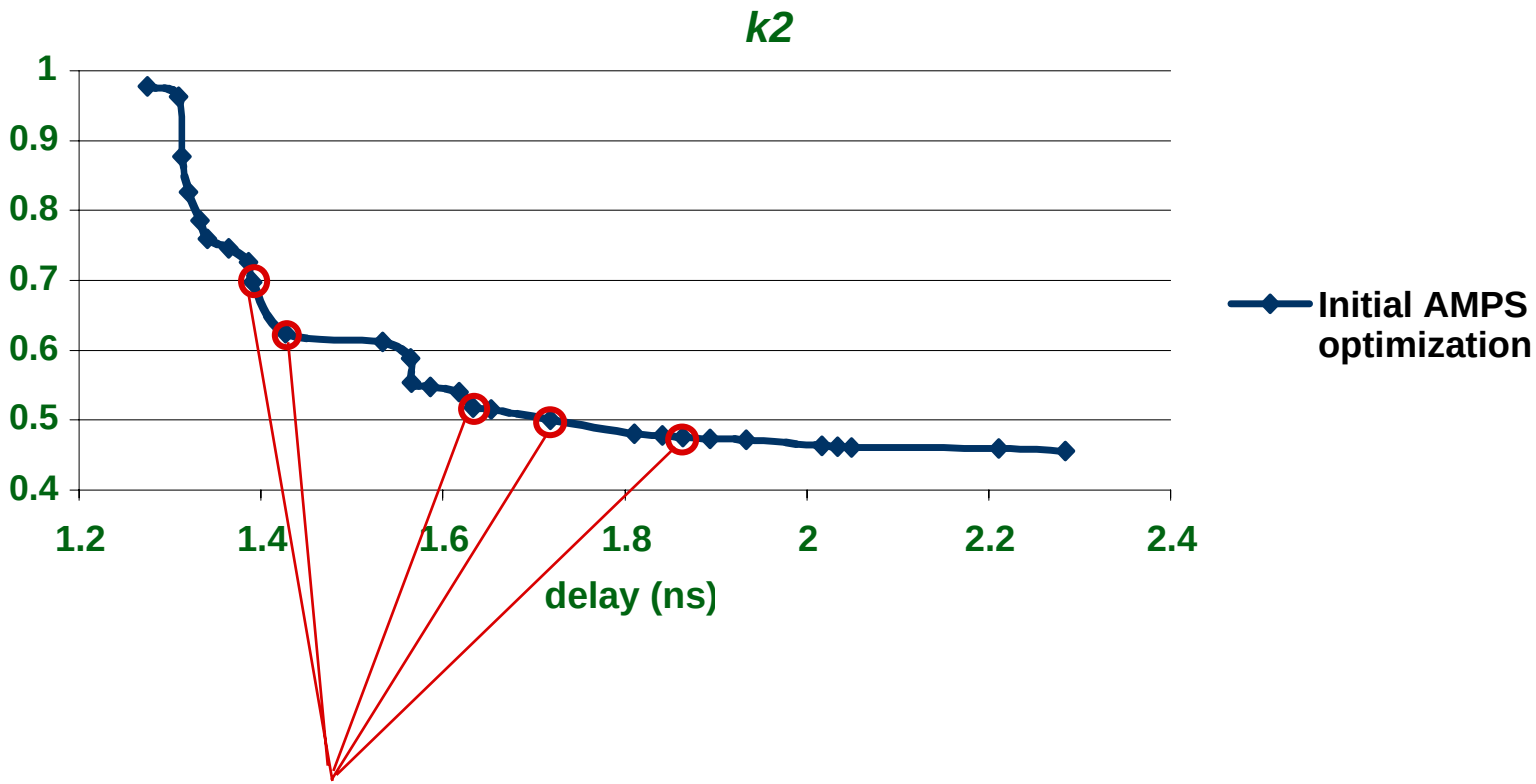
Timing Closure Design Flow (cont'd)

7. Initial Power/Delay optimization using AMPS
8. "Optimized" P-D curve extraction



Timing Closure Design Flow (cont'd)

9. Choose desired points from one or more delay sub-ranges
   - Divide delay range in selected number of sub-ranges (*e.g.* 5 sub-ranges)
   - Choose one point from each sub-range which satisfies specified criteria (*e.g.* min PD)
   - Generate netlist for each of the (*e.g.* 5) designs
   - Netlists corresponding to same synthesis runs differ only with respect to cell sizes

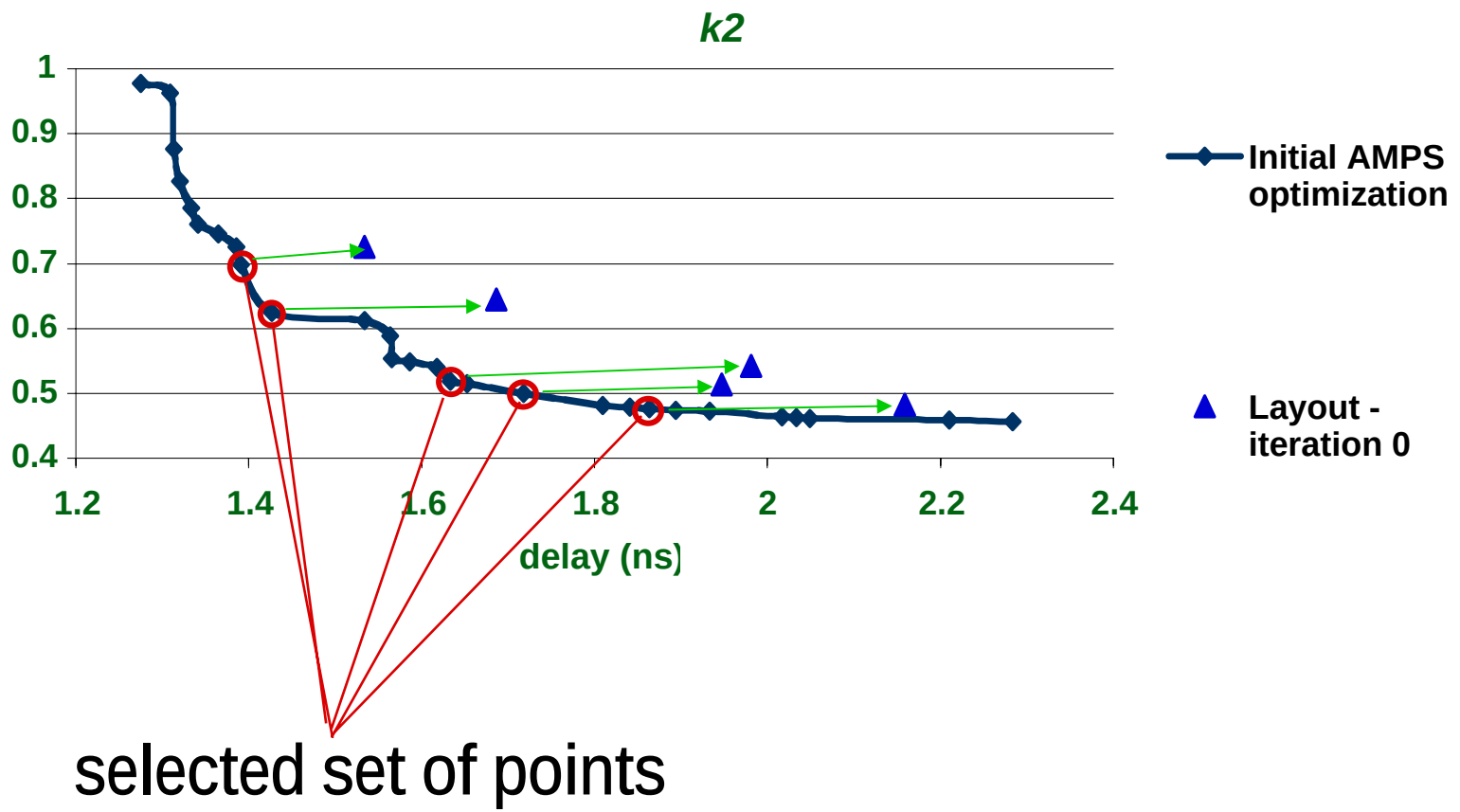


selected set of points

Timing Closure Design Flow (cont'd)

10. Extract needed cell layouts from library
11. High quality (iTools) placement run for each point
12. iTools routing (variable die)
13. Accurate hierarchical parasitic (RC) extraction using Assura RCX or Star RCXT
14. Logical-effort-based clock-tree sizing
15. Detailed skew measurements to the actual FF loads, including distributed RC parasitics
16. Delay/power/area measurements for each generated layout
17. If converged (usually need 1 or at most 2 iterations), stop; else continue

Example - Initial Layout

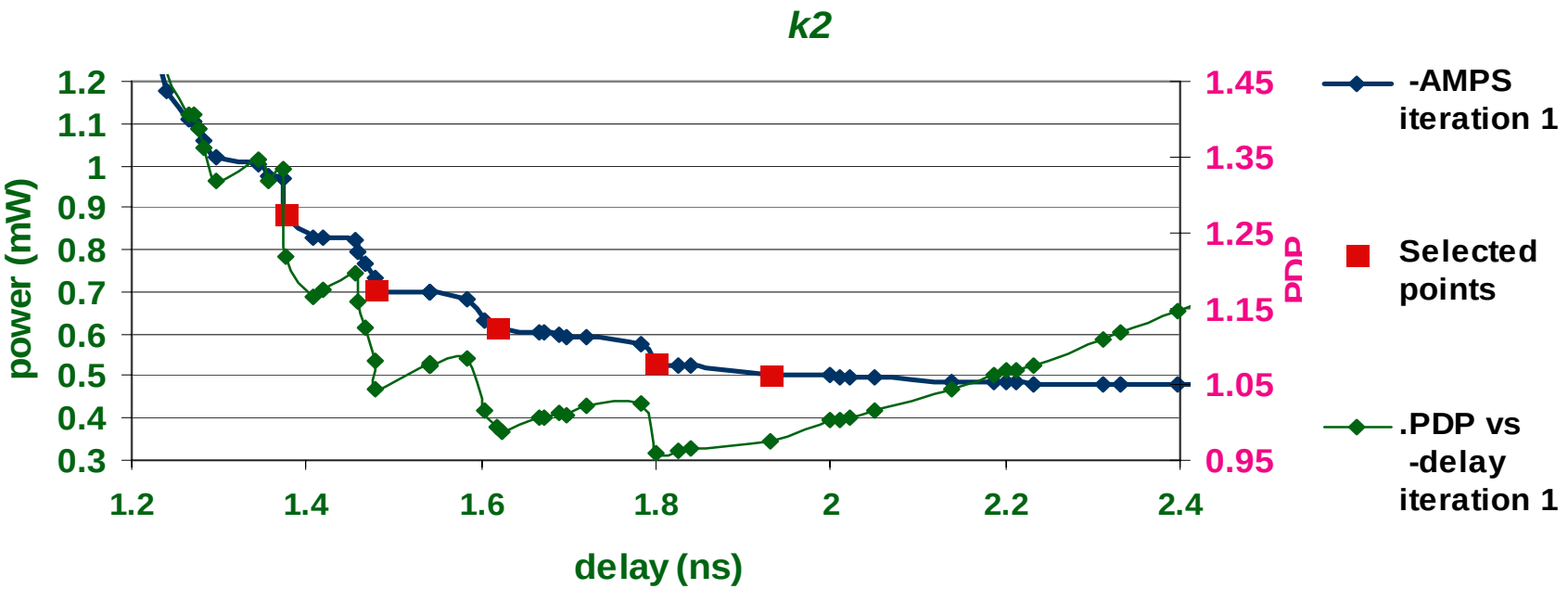


Timing Closure Design Flow (cont'd)

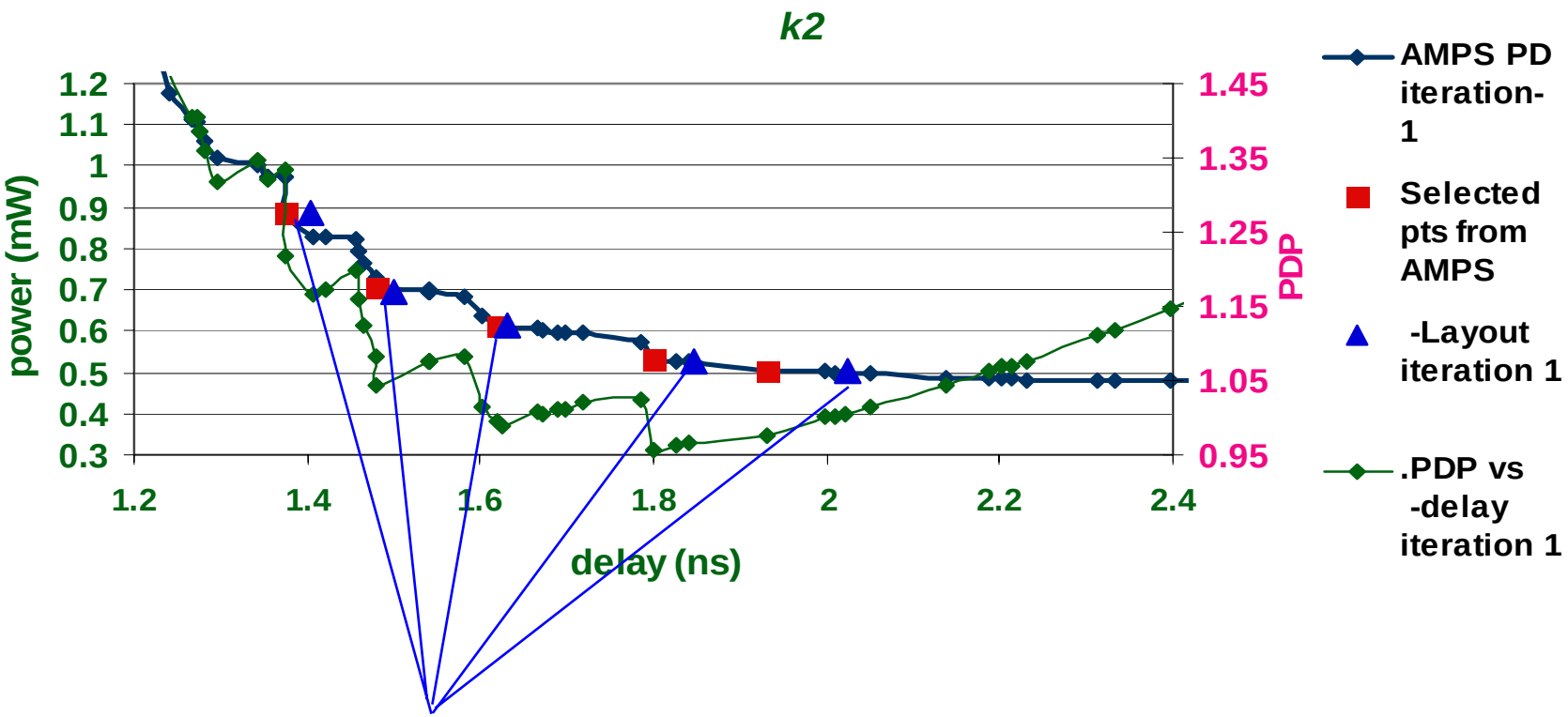
18. New AMPS optimizations for one or more layout points
19. "Optimized" curve extraction over all optimizations
20. Re-execute steps 9-17 where step 11 is now an ECO-based placement run (iTools)

9. Choose desired points from one or more delay sub-ranges
10. Extract needed cell layouts from library
11. High quality (iTools) placement run for each point
12. iTools routing (variable die)
13. Accurate hierarchical parasitic (RC) extraction using Assura RCX or Star RCXT
14. Logical-effort-based clock-tree sizing
15. Detailed skew measurements to the actual FF loads, including distributed RC parasitics
16. Delay/power/area measurements for each generated layout
17. If converged (usually need 1 or at most 2 iterations), stop; else continue

Selected Points after AMPS Iteration 1

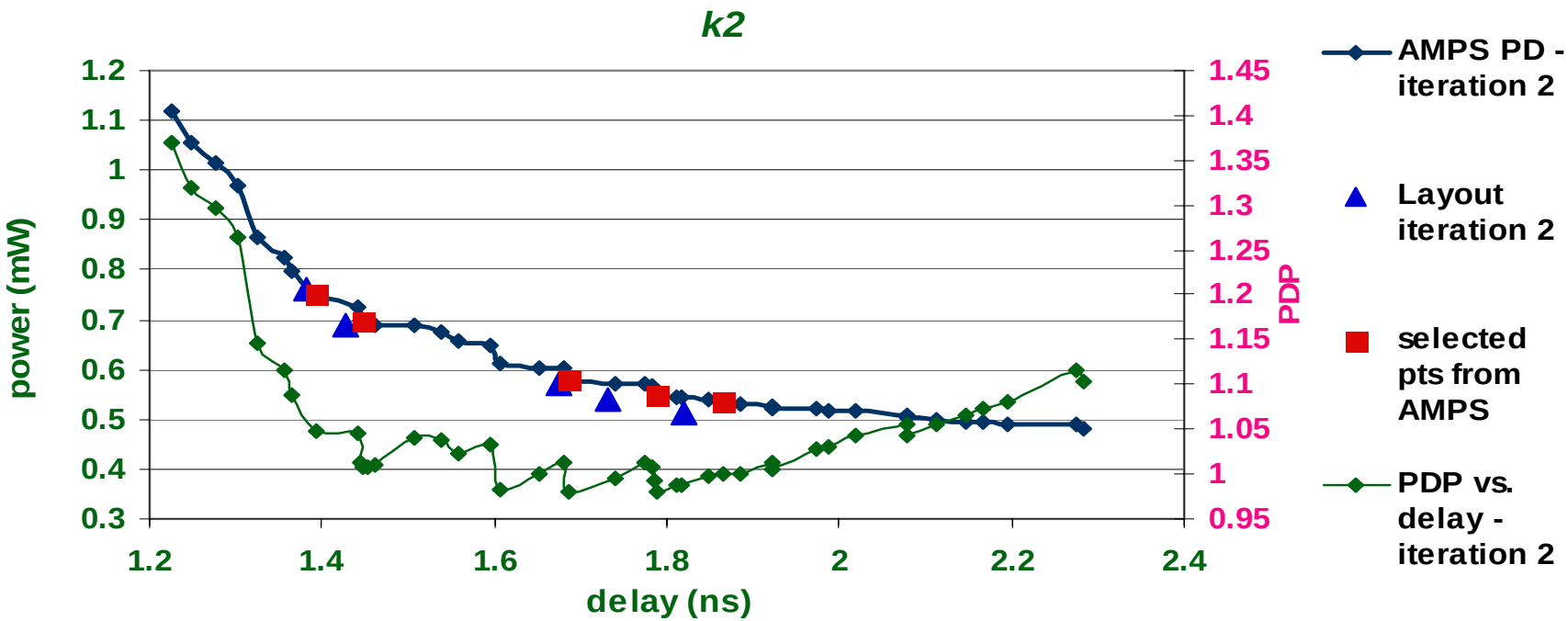


P vs. D after Layout Iteration 1

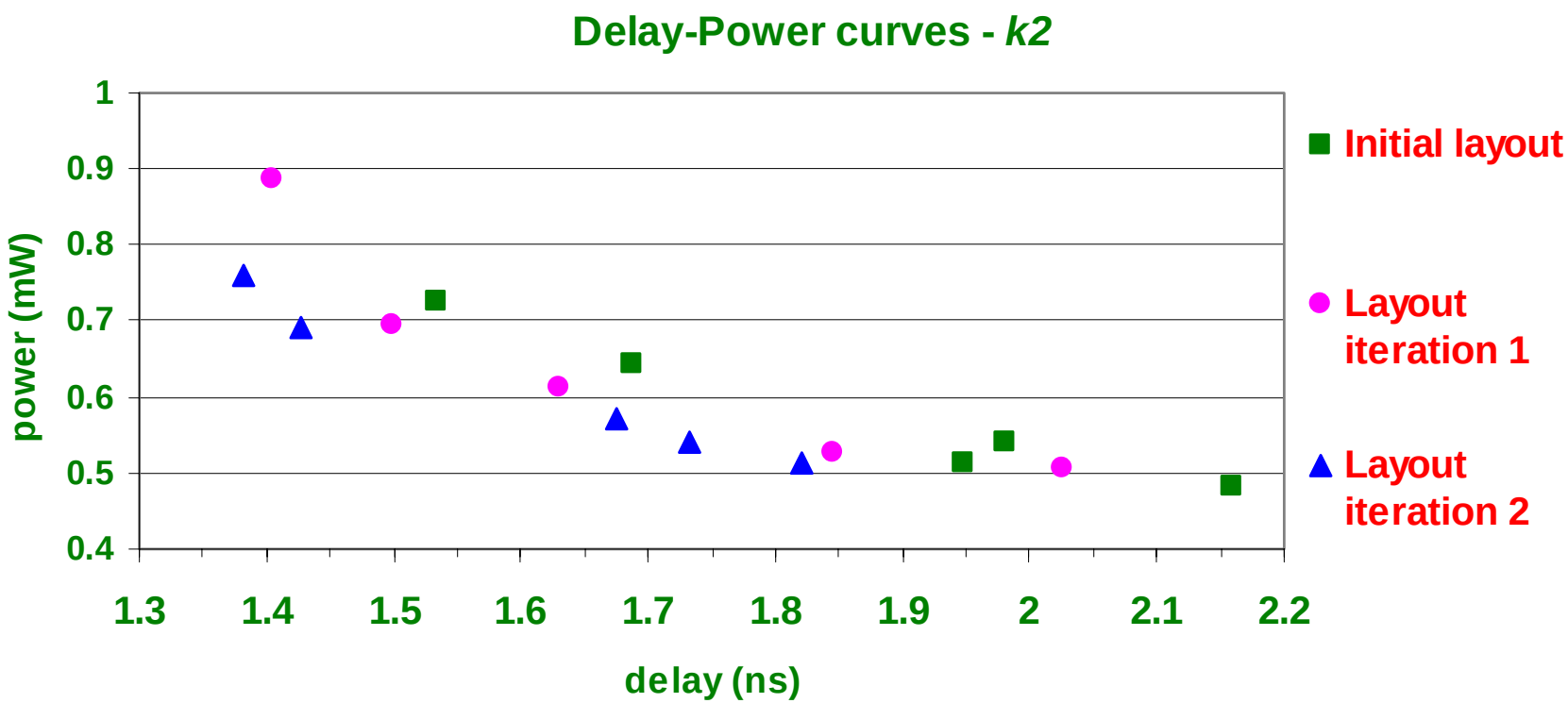


Selected points after layout

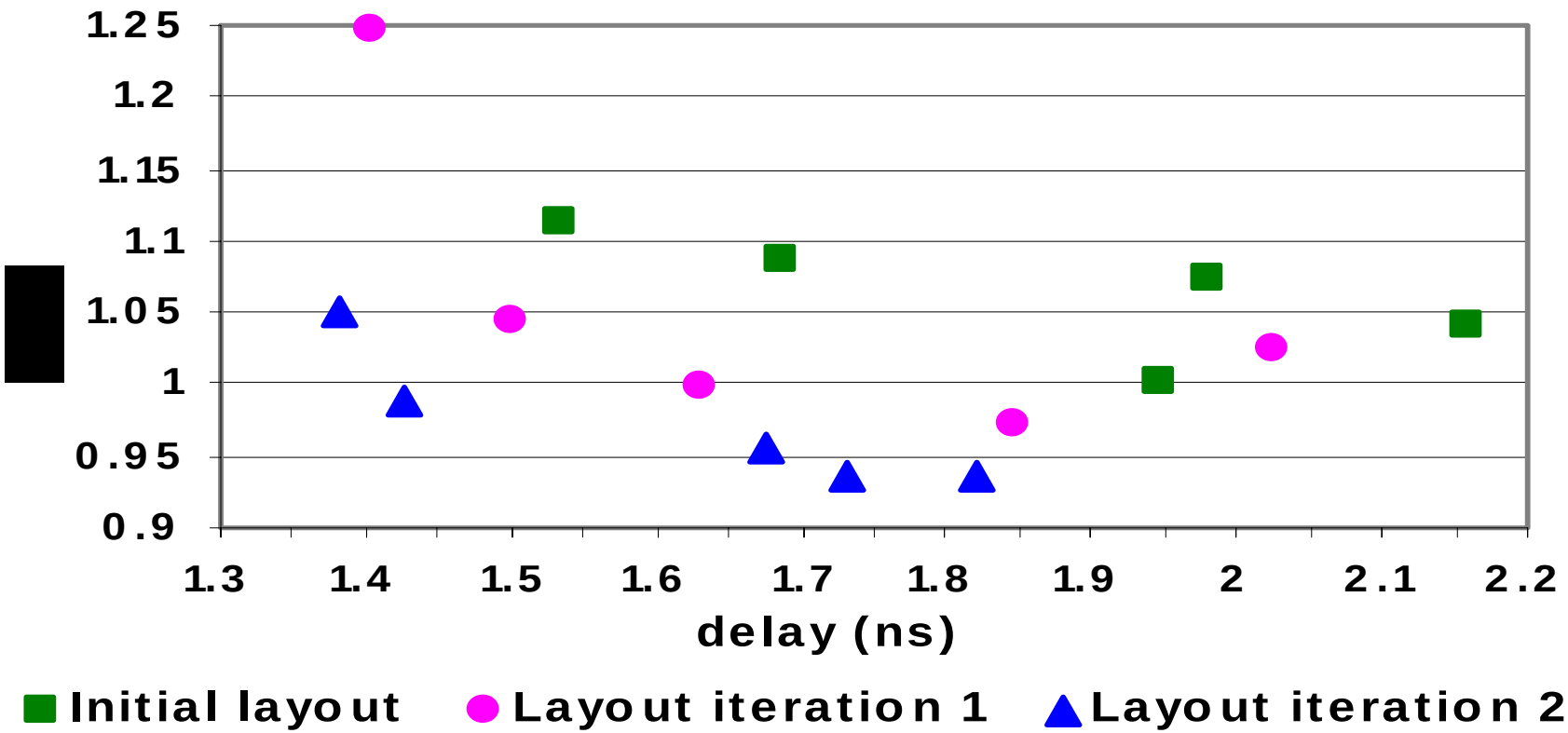
P vs. D after Layout Iteration 2



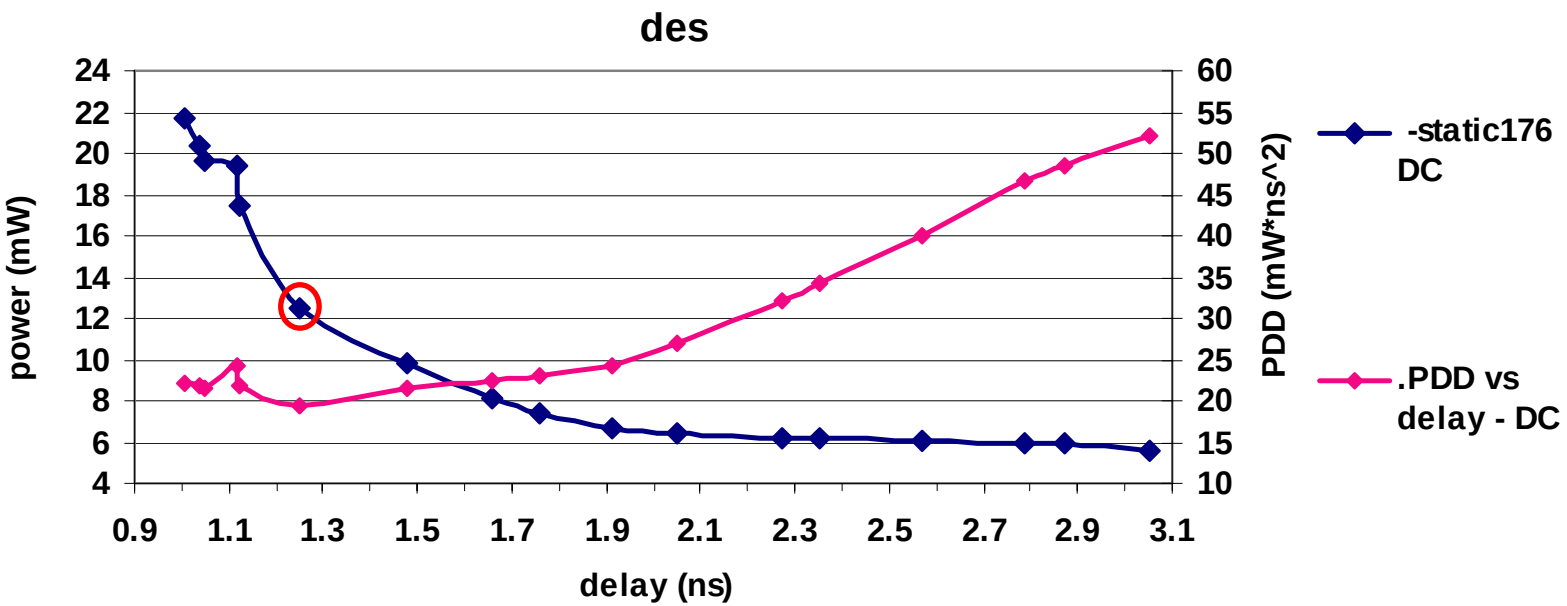
Layout Points after each Iteration



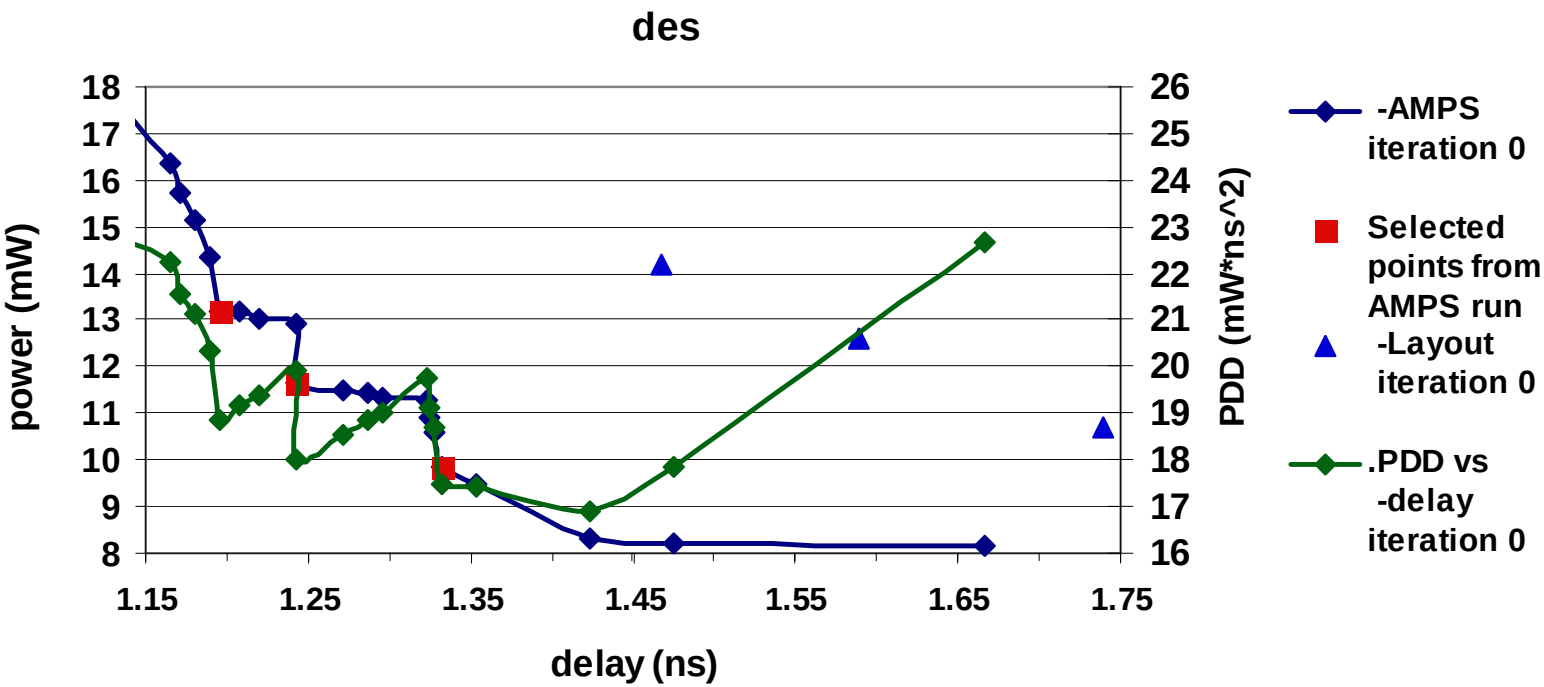
Improvement in PDP for k2 Benchmark



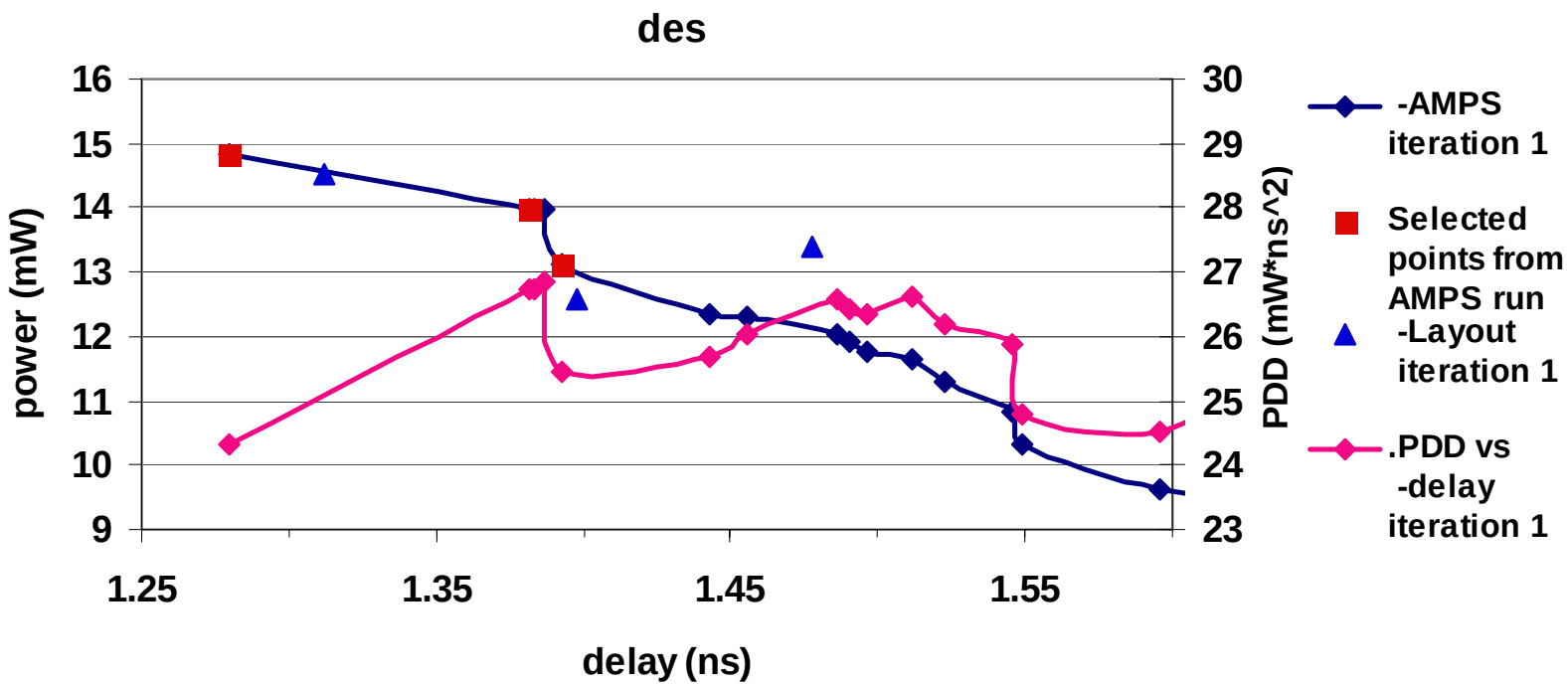
Benchmark *des*



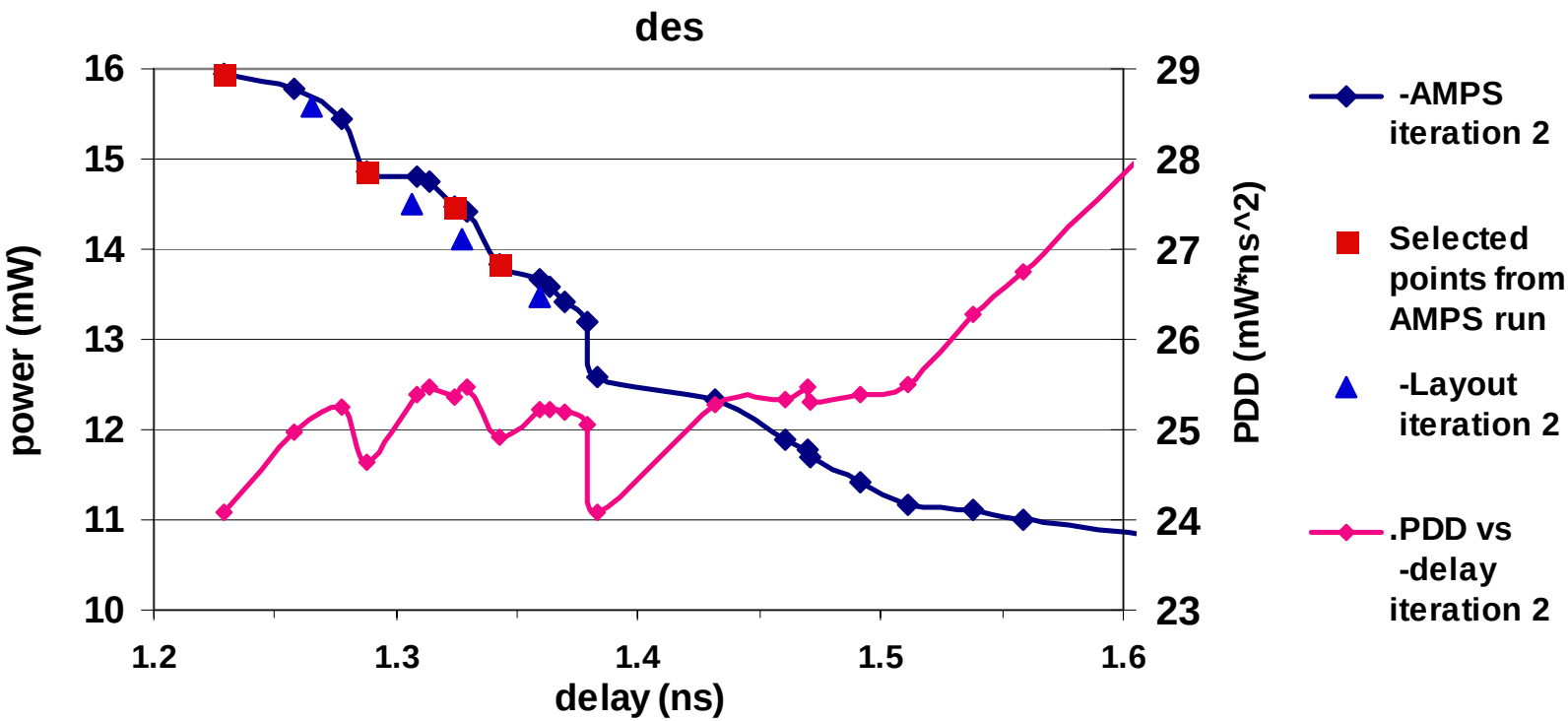
Initial Layout - *des*



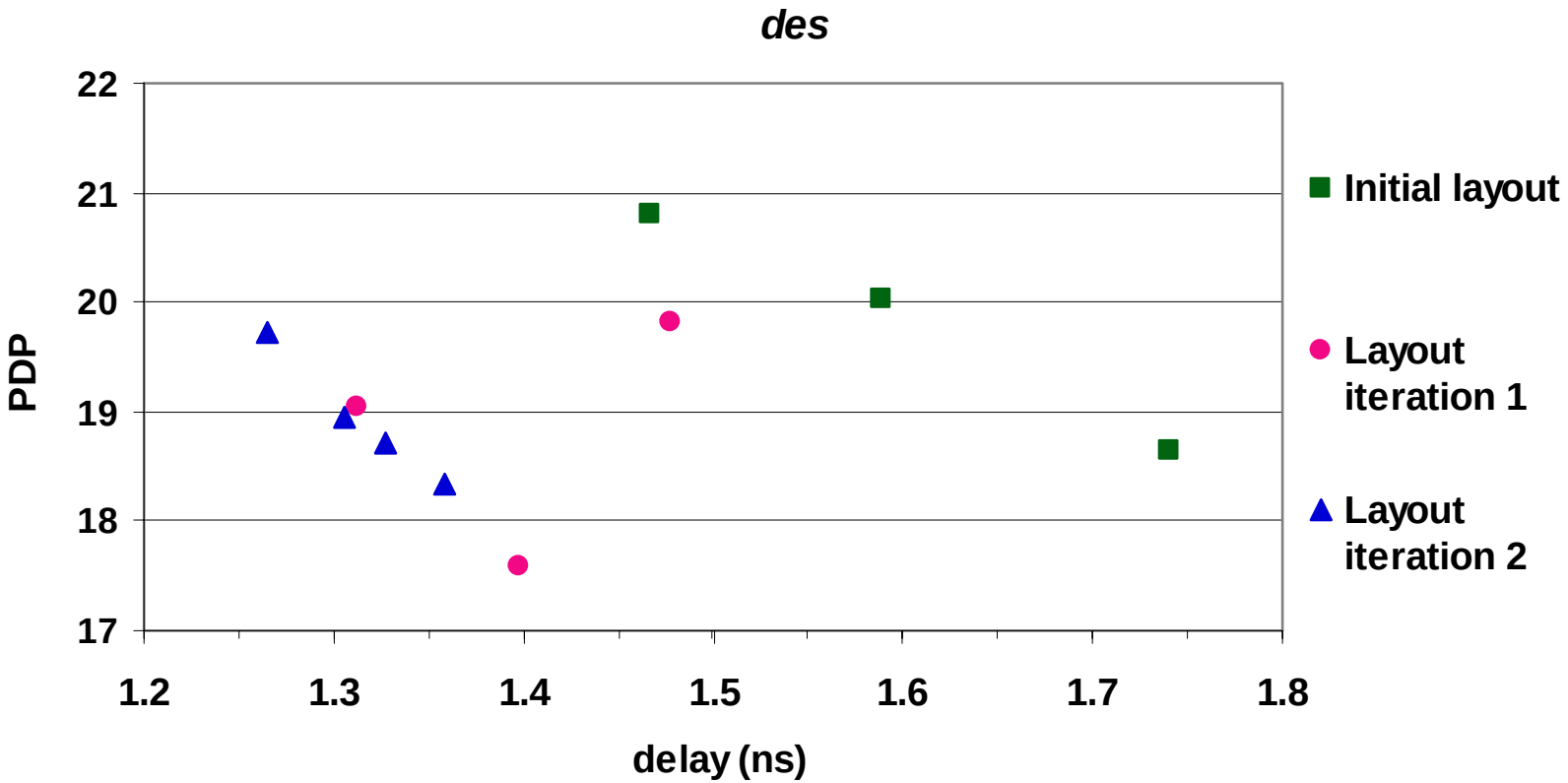
Iteration 1 - *des*



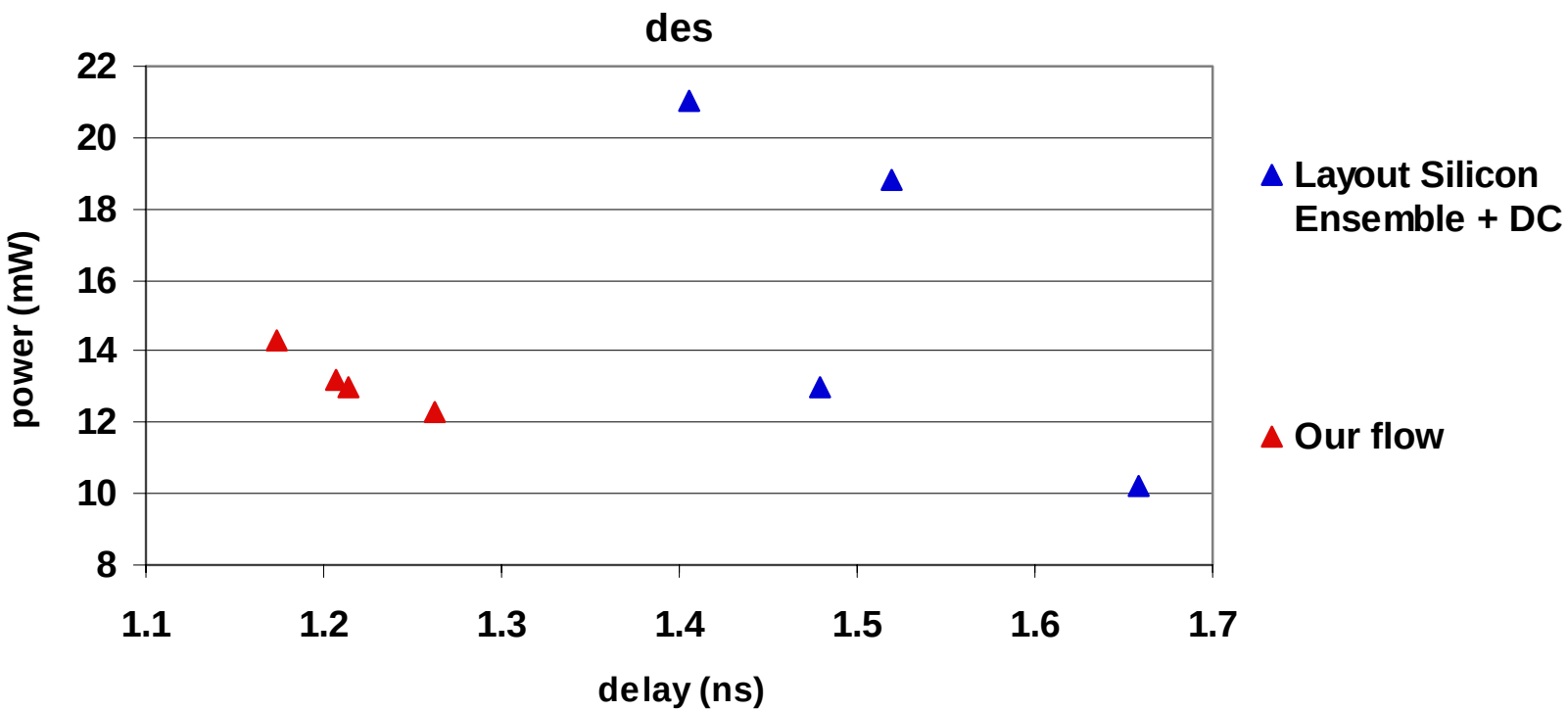
Iteration 2 - *des*



PDP vs. Delay

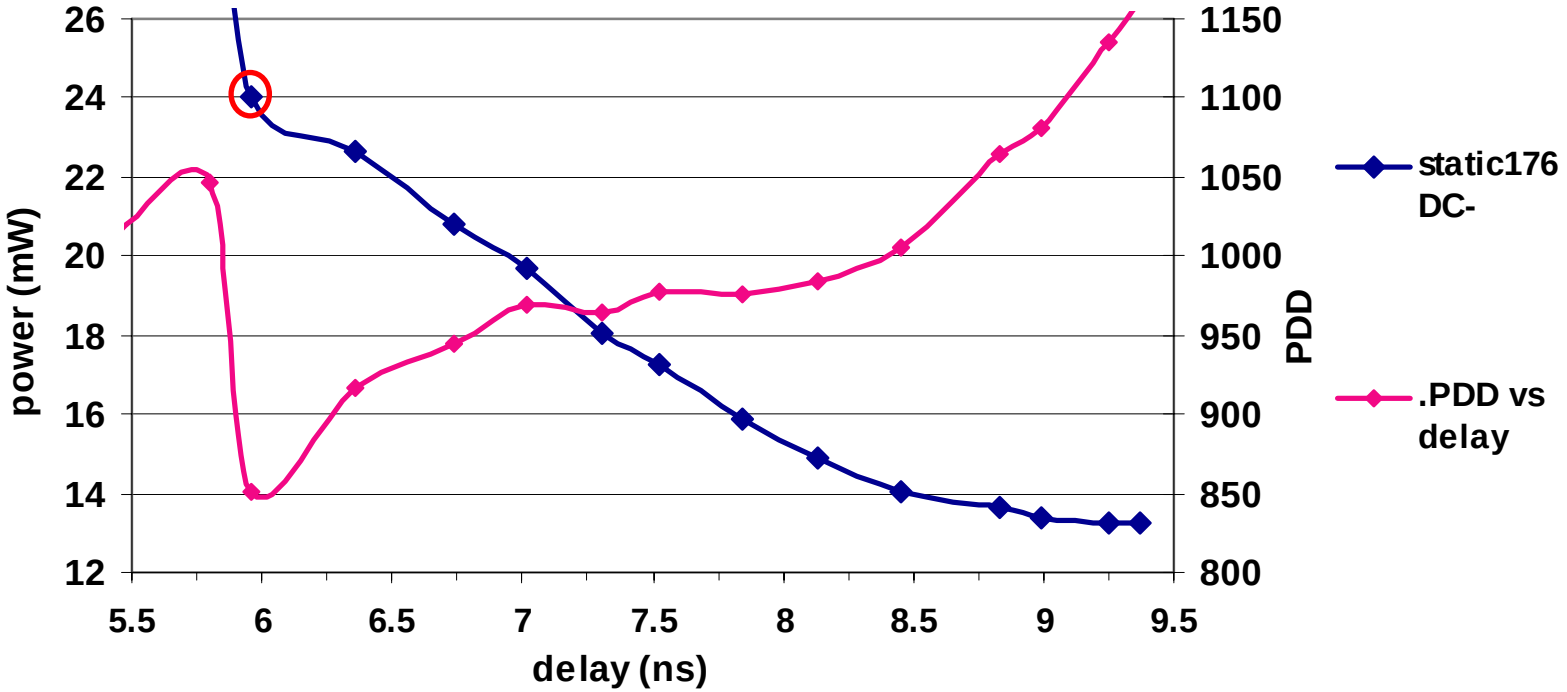

des
22
21
20
19
18
17
PDP
1.2
1.3
1.4
1.5
1.6
1.7
1.8
delay (ns)
Initial layout
Layout iteration 1
Layout iteration 2

DC + Artisan + SE vs. Our Flow



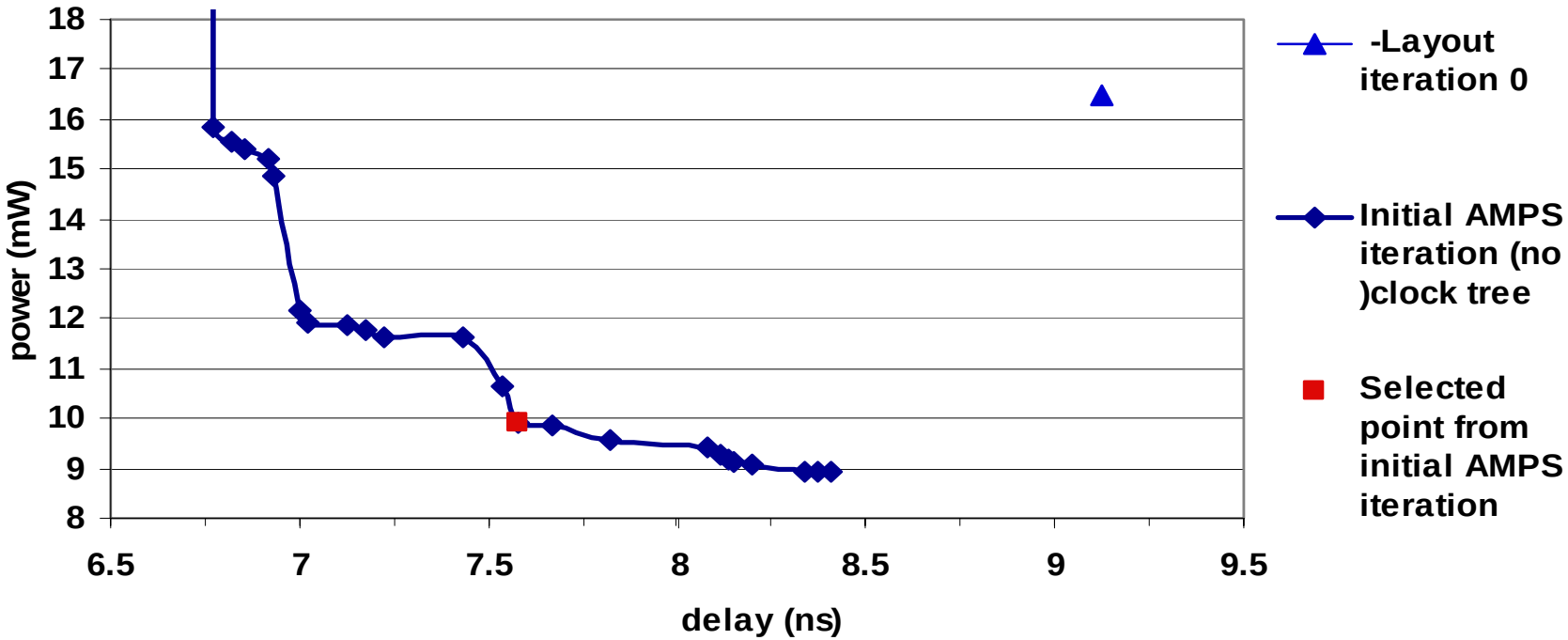
32b FIR after DC

- 32b FIR Filter



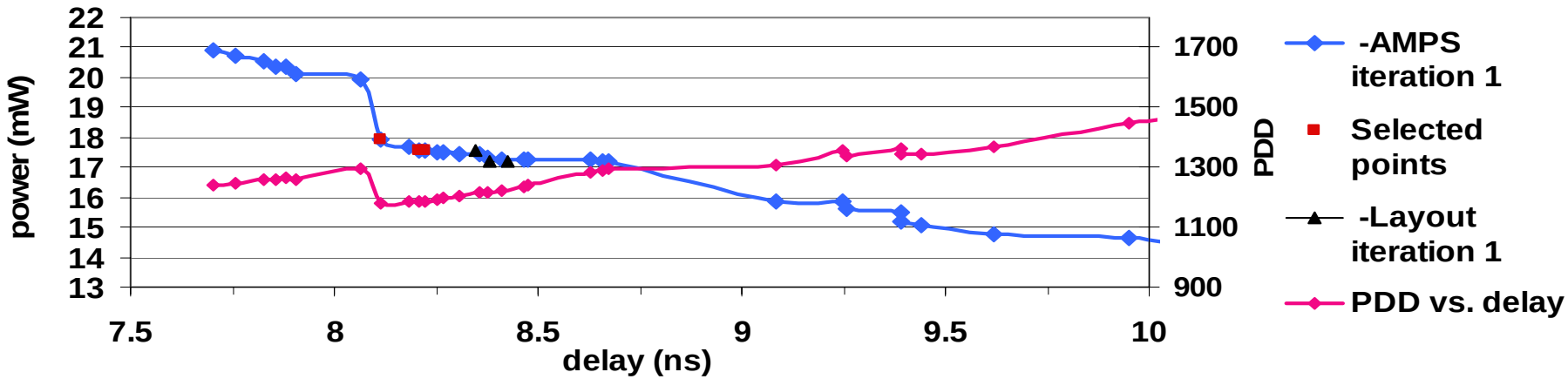
32b FIR: Initial Layout

- 32b FIR Filter



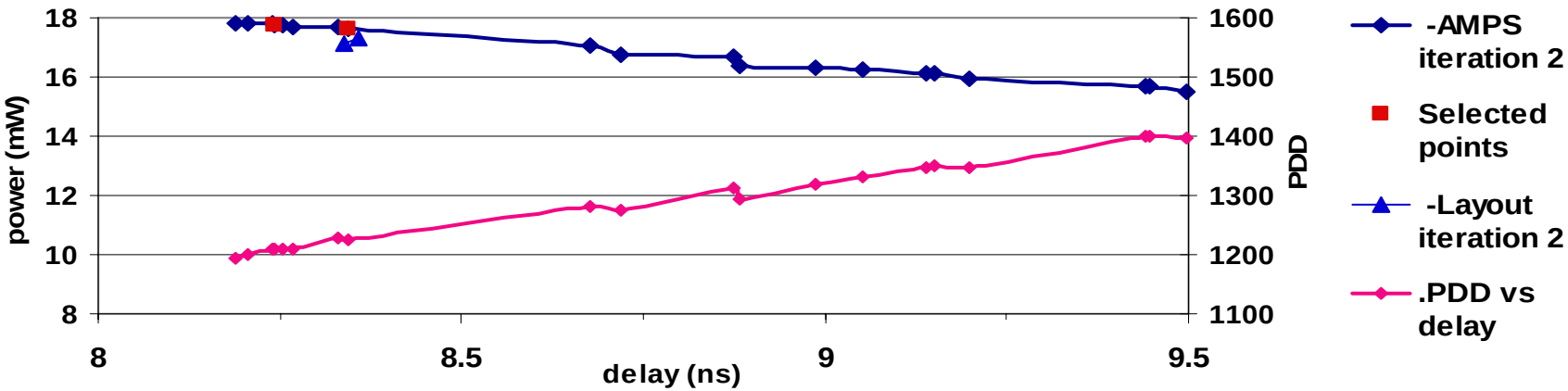
32b FIR – Iteration 1

- 32b FIR Filter



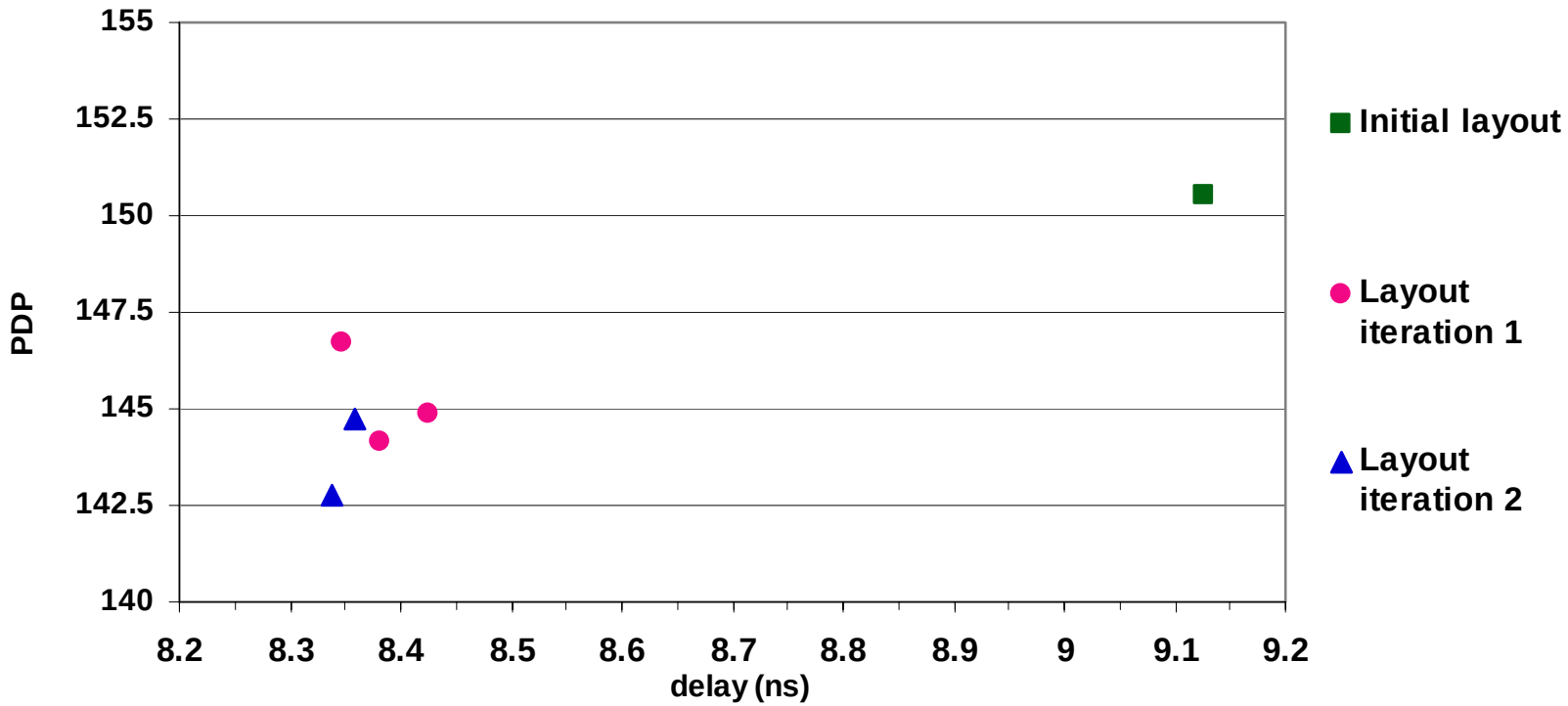
32b FIR – Iteration 2

- 32b FIR Filter



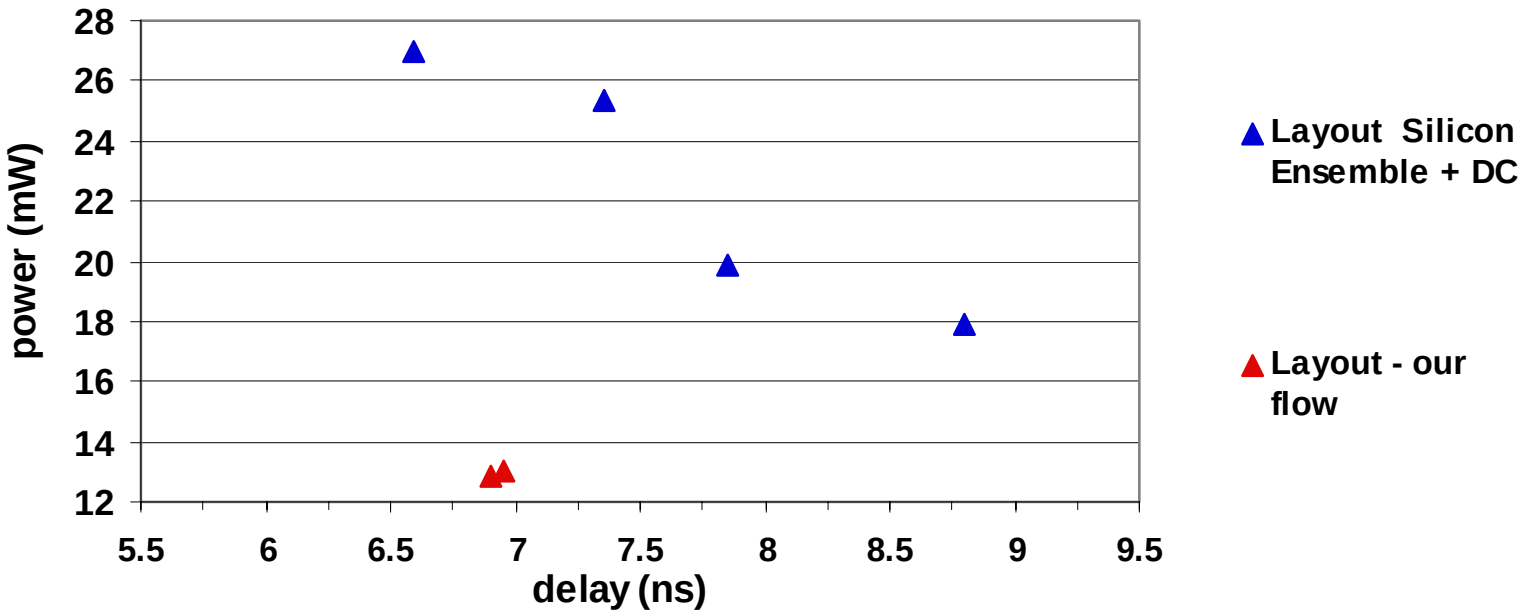
32b FIR: PDP vs. Delay

- 32b FIR Filter



DC + Artisan + SE vs. Our Flow

- 32b FIR Filter



Energy-Delay-Squared Results

Design	Minimum EDD					
	Initial		iteration 1		iteration 2	
K2	1.7	1	1.6	9%	1.4	21%
C5315	17.8	1	16.6	8%	14.5	23%
C7552	29.7	1	25.5	17%	23.2	28%
32-stage FIR	1374	1	1208	14%	1190	15%
Avg improv.		1		12%		22%

Conclusion

- Design flow from Verilog/VHDL to layout mitigates the timing closure problem, while requiring no timing driven placement or routing tools
- Timing issues are confined to the cell sizer, allowing the placement algorithm to focus solely on wire lengths, resulting in superior layout densities and much lower energy (power)
- The key enablers are:
  - Optimal library composition (power efficient gates only)
  - Huge number of drive strengths and beta ratios
  - Effective scheme to generate energy (power) vs. delay plots
  - Lowest wire length placement
  - Variable die routing that allows each net to be routed in the shortest possible length and guarantees 100% routing completion in the smallest possible area
  - Integrated cell placement, routing, gate sizing, and clock tree insertion
  - Effective incremental (ECO) placement that preserves net lengths from one iteration to the next (made possible by the “stability” of the variable die router)

Acknowledgements

- We are very grateful for the financial support provided by:
  - Intel Corporation
  - Boeing/DARPA (MSP program)
  - MARCO/C2S2
  - National Science Foundation (NSF)
  - NSF Center for the Design of Digital and Analog ICs (CDADIC)
  - SRC (early on)
  - Texas Instruments
  - AMD