

Parallel Standard Cell Placement on a Cluster of Workstations



Nael B. Abu-Ghazaleh

Patrick H. Madden

Faris H. Khundakjie

Mehmet Can Yildiz

Computer System Research Lab

BLAC CAD

State University of New York at Binghamton

Computer Science Department

<http://www.cs.binghamton.edu/research.html>

Outline

- ◆ The Standard Cell Placement Problem
 - * Partitioning-based Approaches
- ◆ Feng Shui
 - * Global Optimization * Algorithm Formulation
- ◆ Need for Parallelism
 - * Quality-Speedup-Cost Tradeoffs
- ◆ Parallel Implementation`
 - * Partitioning * Branch & Bound Reordering
- ◆ Experiments
- ◆ Conclusions



Solution Space

◆ Simulated Annealing

Algorithm

- Start with very high “temperature” and random placement.
- Perturb by randomly moving or swapping elements.
- If new quality is better, accept perturbation. Otherwise, accept with probability.
- Cool off

Discussion

- Can produce high quality results only if the cooling schedule and ratio of moves to swaps are designed carefully. Success has been limited to few examples such as TimberWolf and IBM.
- Large runtime and questionable obedience to parallelism.

Solution Space (Cont.)

◆ Force Directed or LP

Algorithm

Form a system of equations and solve iteratively.

Discussion

Popular in commercial tools and a compromise between quality and runtime.

◆ Partitioning based (will revisit)

Algorithm(s)

A variety exists. Basically exploit the concept of divide-and-conquer to arrive at cell locations by n-way partitioning of regions into smaller and smaller ones.

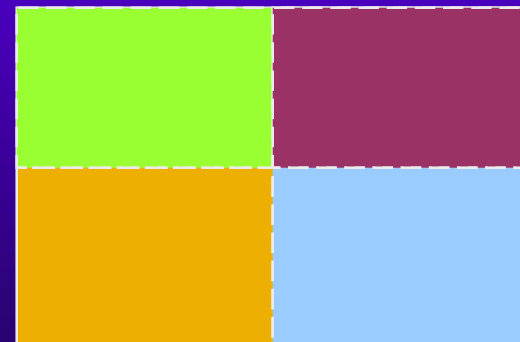
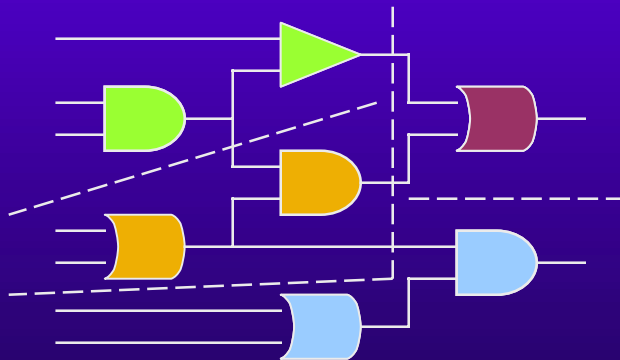
Discussion

Much faster than other approaches, offers freedom in algorithm formulations to optimize for quality/time, and inherently supports parallelism.

Feng Shui

- ◆ State-of-the-art partitioning based tool.
- ◆ Unlike traditional approaches, it achieves globally optimized placement in fast runtimes using *Iterative Deletion (ID)*.

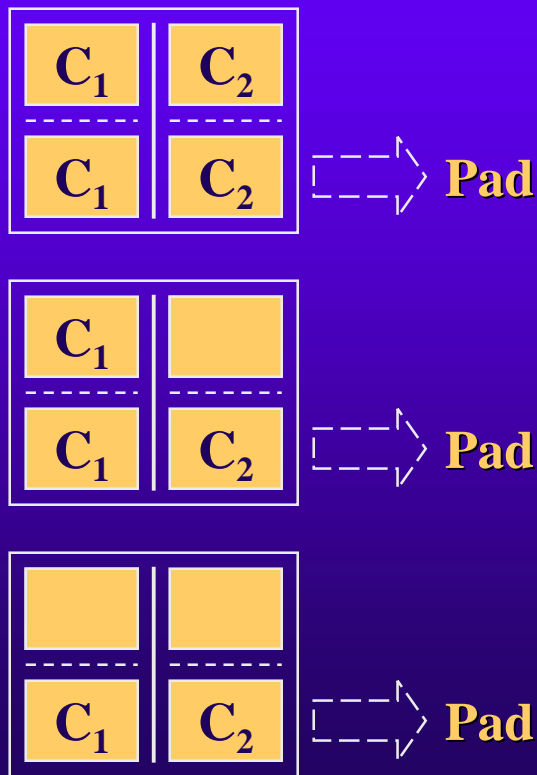
Recursive Bi-partitioning



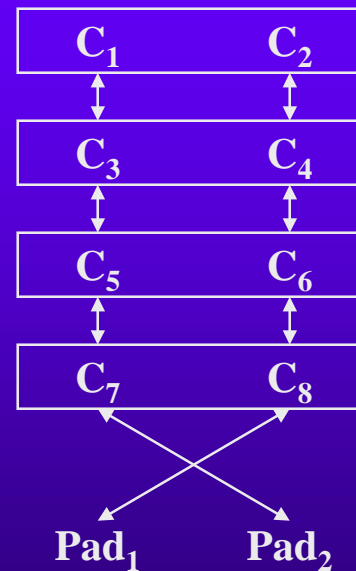
Feng Shui – Iterative Deletion

- ♦ Locally optimized solutions may not necessarily be optimal for the whole circuit.

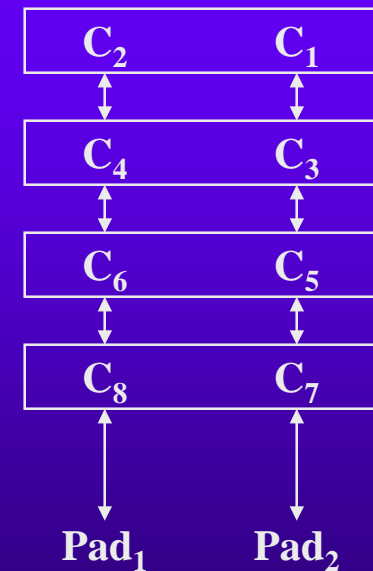
Iterative Deletion



Localized



Optimal

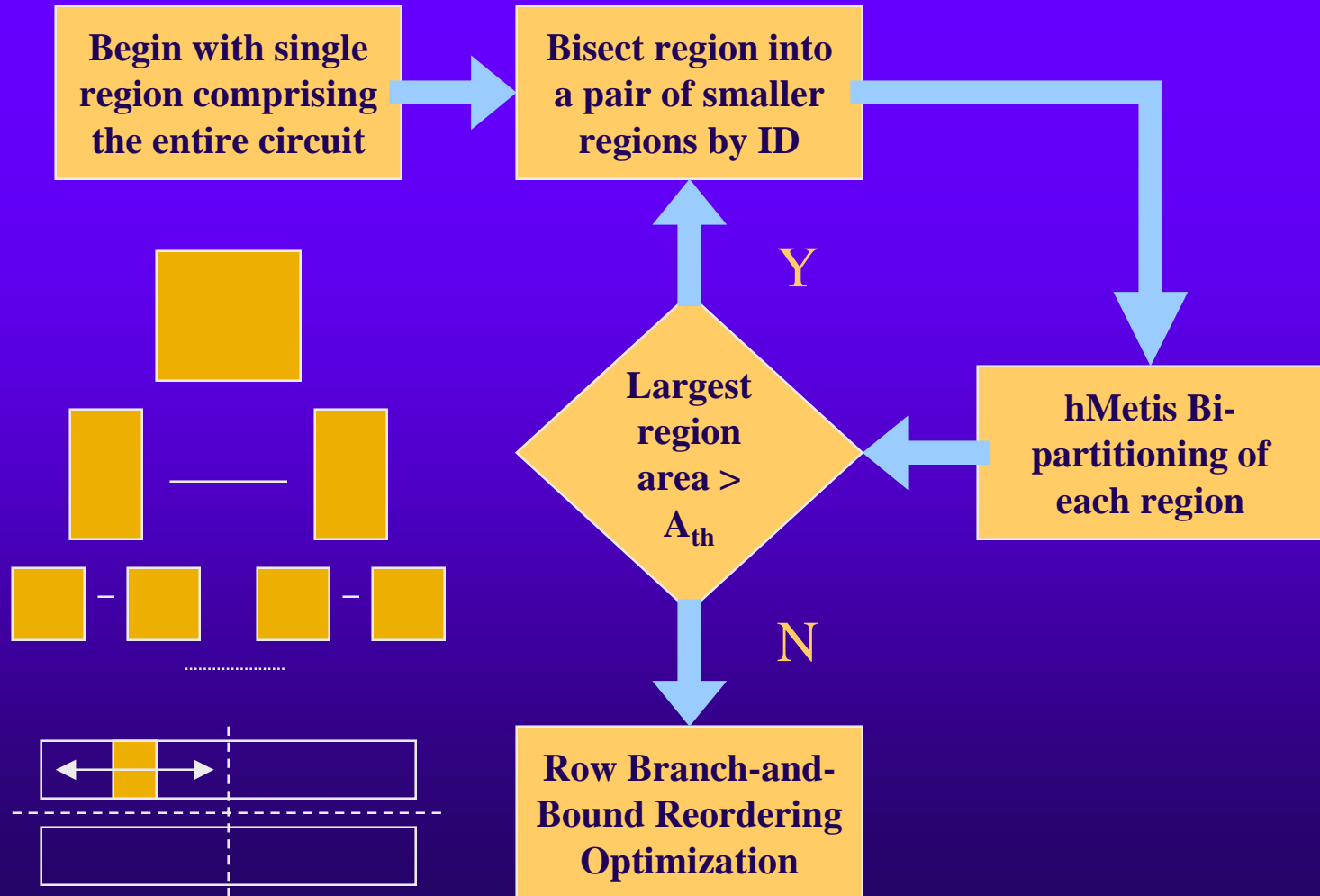


Netlist

$C_1 \leftrightarrow C_3 \leftrightarrow C_5 \leftrightarrow C_7 \leftrightarrow \text{Pad}_1$

$C_2 \leftrightarrow C_4 \leftrightarrow C_6 \leftrightarrow C_8 \leftrightarrow \text{Pad}_2$

Feng Shui – Block Diagram





Need for Parallelism

- ◆ The ever increasing size of circuits presents a growing challenge for automation tools.
- ◆ Parallelism can allow for a new class of algorithm formulations that makes use of more processors perhaps not only to gain speedup but also to achieve better quality.
- ◆ More computationally expensive models would seem more feasible, for example time driven placement.

Tradeoffs

◆ Quality-Speedup

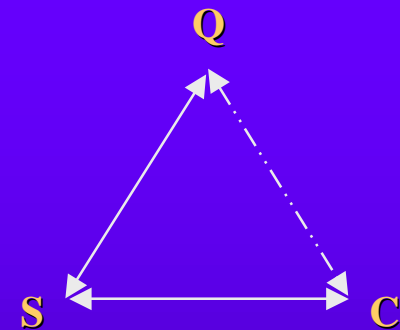
Traditionally quality degrades with higher degree of parallelism as dependencies and order become more difficult to determine.

◆ Speedup-Cost

Trivial observation. More processing power is not for free. Clusters are one of the cheapest options yet they provide significant speedup.

◆ Quality-Cost

More complex high quality algorithms benefit from more expensive faster machines with larger memory.



Parallel Implementation

- ◆ A number of experiments showed that hMetis bi-partitioning and B&B reordering form the bulk of run time. Iterative deletion updates contributed very little overhead.
- ◆ The first objective remains the same: quality first then speedup.
- ◆ Clusters offer a wide space of scalable cost-effective parallel processing options.



Parallel Partitioning

◆ Unit Work

The smallest is a target region. This is the minimum the hMetis partitioning engine can operate on.

◆ Computational Model

ID needs to update all costs across all regions. In order to preserve global optimization a master-slave model is used to process the growing list of regions each pass.

◆ Load Balance

hMetis produces balanced cuts and work units are distributed evenly by the master.

◆ Communication Model

- Round robin unit/message model
- MPI Scatter-Gather for (much faster) single message carries all regions to a slave.

Parallel Partitioning - Analysis

◆ Assumptions

- Partitioning engine runs in linear time.
- Homogeneous distribution of circuit elements.

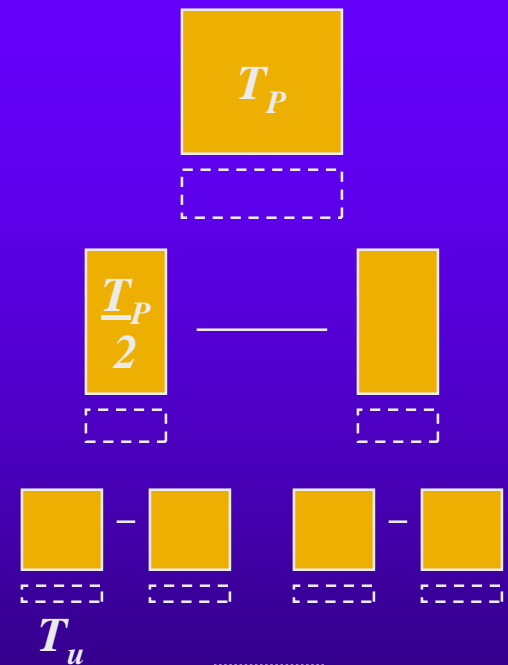
◆ Serial Implementation

$$\underbrace{\frac{T_P}{n}}_{\text{pass}} * n * \underbrace{\log_2 N}_{\text{no. passes}} = T_P * \log_2 N$$

◆ Parallel Implementation

- For the first $(\log_2 P) + 1$ passes, $n \leq P$:

$$\sum_{i=0}^{\log_2 P} \frac{T_P}{2^i}$$





Parallel Partitioning - Analysis

-For the rest, $n > P$, $\text{ceil}(n/P)$ regions per processor:

$$\underbrace{\sum_{i=\log_2 P+1}^{\log_2 N-1} \underbrace{\text{ceil}\left(\frac{n}{P}\right)}_{\text{work}} * \underbrace{\frac{T_P}{n}}_{\text{pass}}}_{\text{total}} \approx \frac{T_P}{P} * (\log_2 N - \log_2 P - 1)$$

-The second part dominates the time when $N \gg P$ or $\log_2 N \gg \log_2 P + 1$ making the total parallel computation time $(T_P/P) * \log_2 N$. Hence, the intuition is linear speedup from the serial part, but ...

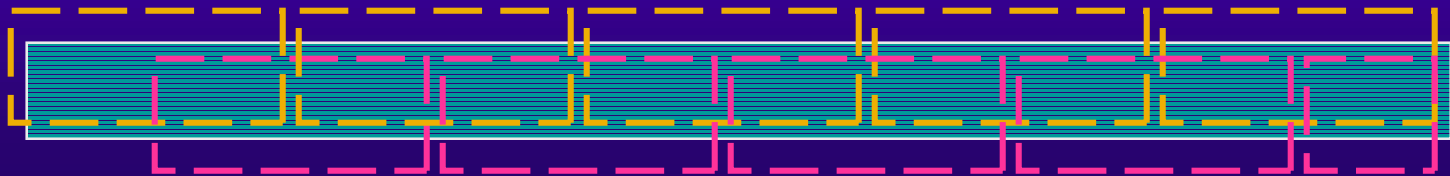
♦ Communication and Update Times

$$\sum_{i=0}^{\log_2 N-1} 2^i * T_u = (N-1) * T_u$$

B&B Reordering

♦ Optimization

- A window with size x cells is slid across the row in steps of half its size and all permutations are considered. The best cost is memorized to skip worse permutations and determine the best exact location of a cell.
- The key for quality is to consider each cell twice by overlapping windows.
- Runtime complexity is $x!$. Larger window size produces slightly higher quality but increases runtime profoundly.





Parallel Reordering

- ◆ **Unit Work**

Window (block of cells)

- ◆ **Computational Model**


Serialize processing of overlapping windows. Reordering is done in two passes to preserve quality.

- ◆ **Load Balance**

Distribute blocks evenly among processors

- ◆ **Communication Model**

Same as parallel partitioning



Experiments – Environment

- ◆ Eight Workstations

550 MHz Pentium III

128 MB RAM

- ◆ Myrinet Network

33 MHz LANai 7

33 MHz, 32-bit PCI bus

- ◆ Switched 100 Mbps Ethernet

- ◆ RedHat Linux 6.1 OS

- ◆ Tool Development

C-Language

MPI 1.1 (MPICH – BIP and MPI – Ethernet)

Linked to hMetis 1.5.3 and other libraries

- ◆ MCNC Benchmarks



Quality-Speedup I

Benchmark	Wire Length		Run Time (sec)			
	Sequential	Parallel (ratio)	Sequential	2 Processors	4 Processors	8 Processors
fract	66242	69498.4(1.05)	2.6	7	7	6.9
struct	775636	797647.6(1.03)	41.9	30.5	21.5	16.6
primary1	1053258	1064441(1.01)	18.6	16.9	13.2	11.8
primary2	3747715	3855615.4(1.03)	80.2	57.1	39.2	30.1
biomed	3382200	3514514(1.04)	162	111.8	79.4	64.5
industry2	15629154	16508315(1.06)	359.4	227.2	155.5	117.8
industry3	45088174	47757192.8(1.06)	538.6	338.5	218.1	157
avqsmall	6041243	6626165.6(1.1)	925.1	746.8	598.9	566.3
avqlarge	6344469	7053757.8(1.11)	1006	788.2	645.5	603.9
golem3	88959019	92830917(1.04)	3235.9	2191.2	1243.1	918.1

Table 2: Wire Lengths and Run Times for the Parallel Implementation Using Scatter/Gather on Myrinet

- ◆ Quality is independent of degree of parallelism (no. processors).
- ◆ Maximum quality degradation is 6 % with exceptions.
- ◆ Best results on the largest benchmark.



Quality-Speedup I (Cont.)

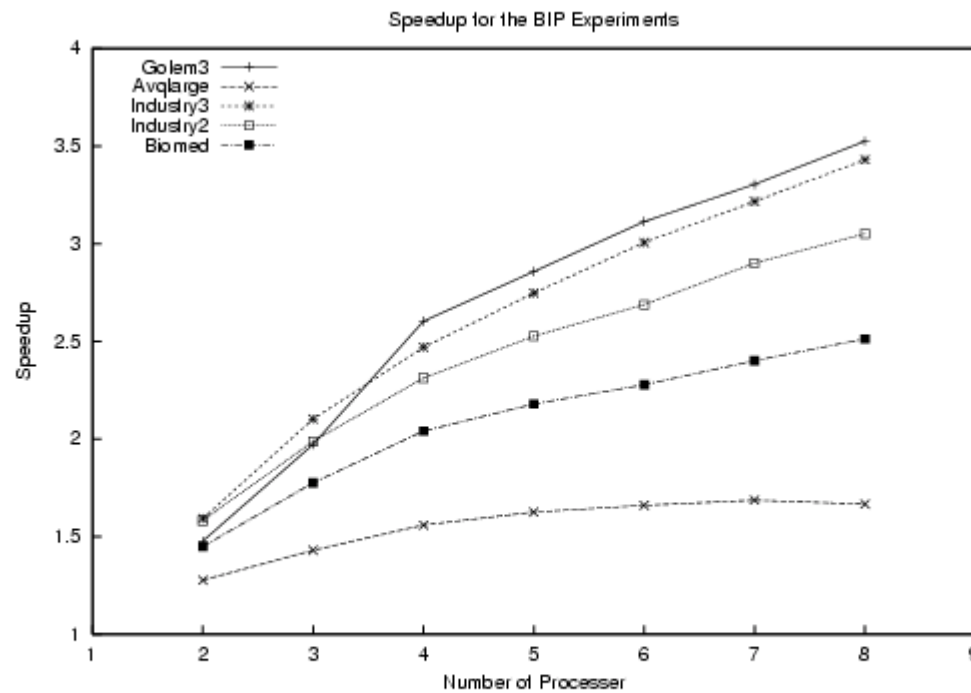
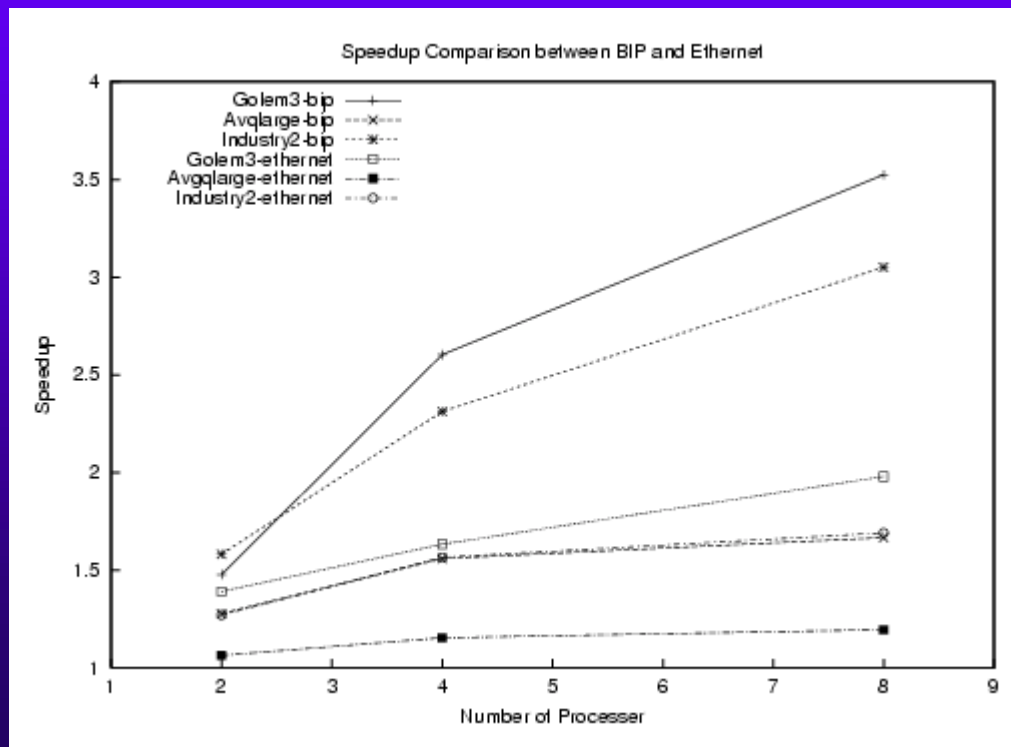


Figure 7: Speedup for the Myrinet/BIP Experiments

Speedup on Ethernet

Benchmark	Run Time (sec)			
	Sequential	2 Proc.	4 Proc.	8 Proc.
industry2	359.4	283	229.7	212.6 (1.69)
industry3	538.6	402.9	309.1	263.9 (2.04)
avqsmall	925.1	894.6	816	772.9 (1.20)
avqlarge	1006	946.2	872.6	842.3 (1.19)
golem3	3235.9	2328	1983	1635



Can we compare results?

- ◆ Only relative quality from the serial implementation.
- ◆ Why?
 - There has been no standard definition of a “bounding box” wire length. Also row spacing is not standardized and assumptions about routing layers vary.
 - Benchmarks were translated/converted differently to suit tools interfaces and placement limitations.
 - It is almost only fair to compare the speedup of two parallel algorithms when they start from the same serial implementation and use the same processing environment.
 - To appreciate how serious this issue is refer to Prof. Madden “Reporting Standard Cell Placement Results” to appear in *IEEE Trans. on Computer Aided Design*.



Conclusions and Future Work

◆ Contribution

- First to decouple quality from parallelism degree. If affordable, add more processor as long as speedup has not saturated.
- One of few to maintain near serial quality
- Significant speedup given the fact that the implementation is not entirely parallel and more than one pass of each parallel phase is executed sequentially.

◆ Future work includes:

- Improving cost update exchange mechanism among processors.
- Modifying Feng Shui to eliminate sequential dependencies and order of partitioning effects or maintaining those dependencies in the parallel implementation using a “wavefront pipeline” scheme to produce quality as high as the sequential version.



Q & A