GPU-Accelerated VLSI Routing Using Group Steiner Trees

Basileal Imana, Venkata Suhas Maringanti and Peter Yoon

Department of Computer Science, Trinity College, Hartford, CT

VLSI Routing Estimation

- In VLSI design flow, logical design is followed by planning the physical layout which involves:

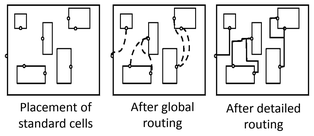
Partitioning → Placement → Routing

- Routing:

Input: Terminal List, location of terminals and pins/ports

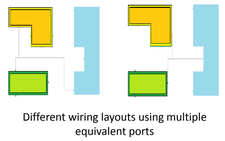
Output: Geometric layout of all nets

Objective: Minimize the total wire length completing all connections without increasing chip area

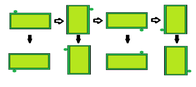


Placement of standard cells | After global routing | After detailed routing

- Conventional procedures of VLSI global routing estimation assume a one-to-one correspondence between terminals and ports.
- In practice, however, each terminal consists of a large collection of electrically equivalent ports, a fact that is not accounted for in layout steps such as wiring estimation. Each module can also be rotated or flipped giving eight possible locations for a given port.



Different wiring layouts using multiple equivalent ports

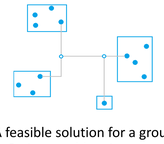


A module is rotated and flipped to induce a group of eight terminal positions

- Tens and thousands of non-overlapping nets may need to be routed simultaneously in large-scale circuit design.

Group Steiner Problem

- Given an undirected weighted graph $G = (V, E)$ and a family $N = \{N_1, \dots, N_k\}$ of k disjoint groups of nodes $N_i \subseteq V$, find a minimum-cost tree which contains at least one node from each group N_i.
- Steiner nodes, i.e., nodes that are not a member of any group N_i, can optionally be used to interconnect the groups. In the figure, the solid dots represent non-Steiner nodes, hollow dots represent optional Steiner nodes and the boxes represent groups.
- The solution to the Group Steiner Problem (GSP) can be used to find an efficient way of connecting a subset of a group of vertices in a graph. Applications of this problem and its variants include routing of VLSI circuits and computer-aided design.



A feasible solution for a group Steiner problem instance

Performance

- All experiments were done on Blue Waters supercomputer which uses a Cray XE6/XK7 system.
- We used several VLSI WRP (Wire Routing Problem) instances from datasets provided by [3] with graph size ranging from 128 to 3,168 vertices.
- We ran strong scalability test by increasing the number of processors while keeping the problem size constant. (Figure 5)
- We also compared our implementation with the best known sequential performance by [4]. The results show our implementation achieves up to 302x speedup for a graph size of 3,168 vertices. (Figure 6)
- Table 1 shows the accuracy of our solutions in comparison with optimum costs from[4].

BLUE WATERS SUSTAINED PETASCALE COMPUTING

Table 1: Accuracy Comparison

Graph	Size	Terminals	Opt. cost [4]	Approx. cost	Error %
wrp3-11	128	11	1100361	1100427	0.006
wrp3-39	703	39	3900450	3900600	0.004
wrp3-96	2518	96	96001172	96003009	0.002
wrp3-83	3168	83	8300906	8302279	0.017

Specifications: Master process runs on a GPU-enabled XK7 compute node while all other slaves run on traditional XE6 compute nodes.

XK7: one AMD 6200 Interlagos CPU, 23GB Host memory; one NVIDIA GK110 "Kepler" accelerator, 6GB memory

XE6: 2 AMD 6276 Interlagos CPUs, 2.3 GHz, 64GB physical memory, 16 cores

Figure 5: A strong scalability test. Also shows comparison with our serial implementation which took 5852.4s for graph size of 2,518 vertices.

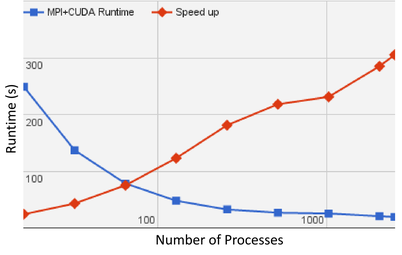
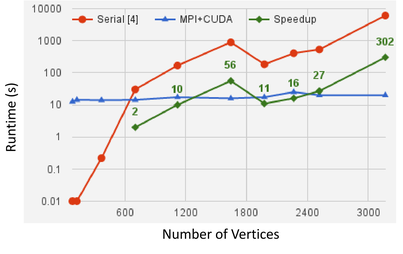


Figure 6: Runtime comparison with [4]



Depth Bounded Tree Approximation

The Group Steiner Heuristic [1]

1. Replace input graph $G = (V, E)$ by its *metric closure*, i.e., a complete graph with vertices V and edge weights equal to the shortest path lengths. (Figure 1)
2. For every terminal vertex v, create a new node v' and a new zero-cost edge (v, v'). v' takes the role of v; v' becomes a port and v a non-port. (Figure 2)
3. Construct a rooted *1-star* tree, i.e., a tree of depth 1 where all leaves are terminal nodes one from each group. (Figure 3)
4. Select intermediate nodes and determine a set of groups that should be connected to each intermediate node to for a *partial-star.* (Figure 4)
5. Combine the *partial-stars* to obtain a *2-star* tree, i.e., a tree of depth 2. (Figure 4)

The *2-star* has to be rooted at the center of the optimal solution-tree. The center, however, cannot be determined ahead of time. Therefore, a Steiner *2-star* is constructed with each node v in G as root r and the minimum-cost tree is determined over all possible choices of r.

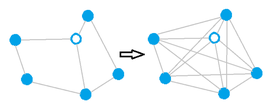


Figure 1: Construction of metric closure

Figure 2: Transformation of input graph

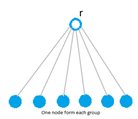


Figure 3: *1-star* tree rooted at r

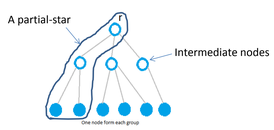


Figure 4: *2-star* tree made of three *partial-stars*

Two-star tree construction

Input: A graph $G = (V, E)$, a family N of k disjoint groups $N_1, \dots, N_k \subseteq V$ and a root $r \in V$
$Approx_2(r) \leftarrow \{r\}$ /*add root to solution*/
$N' \leftarrow N$ /*begin with all groups remaining*/
WHILE $N' \neq \emptyset$ **DO**
  FOR EACH $v \in V$ **DO**
    Sort $M = \{N_1, \dots, N_k\}$ such that $\frac{cost(v,Ni)}{cost(r,Ni)} \leq \frac{cost(v,N_{i+1})}{cost(r,N_{i+1})}$
    Find $j \in \{1, \dots k\}$ that minimizes
      $norm(v) = \frac{cost(r,v) + \sum_{i=1}^{j} cost(v,Ni)}{\sum_{i=1}^{j} cost(r,Ni)}$
    $M(v) \leftarrow \{N_1, \dots, N_k\}$ /*store groups connected to v */
  ENDFOR
  Find v_{min} with the minimum $norm(v)$
  $P \leftarrow (r, v_{min}, M(v_{min}))$ /*partial star with root r, intermediate v */
  $N' \leftarrow N' - groups(P)$
  $Approx_2(r) \leftarrow Approx_2(r) \cup P$ /*add partial star to solution*/
ENDWHILE
Output: A low-cost 2-star $Approx_2(r)$ with the root r intersecting each group N_i

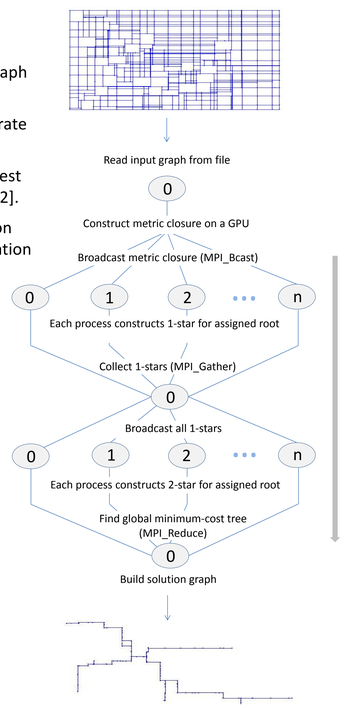
The time complexity of the group Steiner Heuristic is $O(\alpha + |V|^2 \cdot k^2 \cdot \log k)$, where k is the number of groups and α is the time it takes to compute all-pairs shortest paths using Floyd-Warshall Algorithm.

CUDA-Aware MPI-Based Approach

- Our implementation of the Group Steiner Heuristic [1] is based on CUDA-Aware MPI.
- We parallelize the algorithm by exploiting the need to construct rooted 2-star trees with each vertex v in the graph as a possible root.
- All such trees can be constructed independently on separate processes.
- Metric closure is constructed by computing All Pair Shortest Paths using Blocked Floyd-Warshall Algorithm on a GPU [2].
- There is no communication during 2-star tree construction (the most time-consuming step), making our implementation scalable.

CUDA-Aware MPI-based approach

IF master
  $G \leftarrow$ read_graph($filename$)
  $(Mc, P) \leftarrow$ FLOYD_APSP_CUDA(G) /*get metric closure and predecessors matrix*/
ENDIF
BROADCAST(Mc) /*from master*/
$onestar\_all \leftarrow [\,]$
WHILE there is a remaining root r
  $onestar \leftarrow$ build_onestar(Mc, r)
  GATHER($onestar, onestar\_all$) /*to master*/
ENDWHILE
BROADCAST(onestar_all) //from master
$global\_min \leftarrow \{\,\}$
$local\_min \leftarrow \{\,\}$
WHILE there is a remaining root r
  $twostar \leftarrow$ build_twostar(Mc, r ,$onestar\_all$)
  IF cost($twostar$) < cost($local$_min)
    $local$_min $\leftarrow twostar$
  ENDIF
ENDWHILE
REDUCE($local\_min, global\_min$) /*to master*/
IF master
  build_sol_graph ($global\_min, P, Mc$)
ENDIF



- Note the communication pattern when we launch as many processes as the number of the vertices in the graph. If fewer number of processes are launched, round-robin work distribution scheme is used to assign roots.

Discussion and Future Directions

Discussion

- Our implementation outperforms existing serial implementations while producing accurate solutions for WRP problem instances.
- The algorithm is adaptively refined in that it uses several steps to refine the solution and is highly dynamic making it hard to predict the size of the solution before actually computing it. This causes some load imbalance.
- Due to the dynamic nature of the problem, our implementation exhibits some irregular memory access patterns.

Future Directions

- Design more efficient work distributions schemes to decrease the load imbalance.
- Overlap communication with computations where possible.
- Improve the efficiency of memory access pattern.

Acknowledgment

This research was supported by:

- CUDA Teaching Center Program, NVIDIA Research
- Faculty Research Committee and Interdisciplinary Science Program, Trinity College
- Blue Waters Student Internship Program

References

[1] Helvig C.S., Robins, G. and Zelikovsky, A. New Approximation Algorithms for Routing with Multi-Port Terminals. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *19*(10), 1118-1128.

[2] Lund, B. D., and Smith, J. W. A multi-stage CUDA Kernel for Floyd-Warshall. CoRR abs/1001.4108 (2010).

[3] Koch, T., Martin, A. and Voß, S. SteinLib: an updated library on Steiner tree problems in graphs. in Cheng, X. and Du, D.Z. eds. *Steiner Trees in Industry,* Springer US, Berlin, 2001, 285–326.

[4] Polzin, T., Vahdati, S. The Steiner tree challenge: An updated study, *11th DIMACS Implementation Challenge.* Retrieved July 06, 2015, from Princeton University: http://dimacs11.cs.princeton.edu/papers/PolzinVahdatiDIMACS.pdf