
ECE 636

Reconfigurable Computing

Lecture 12

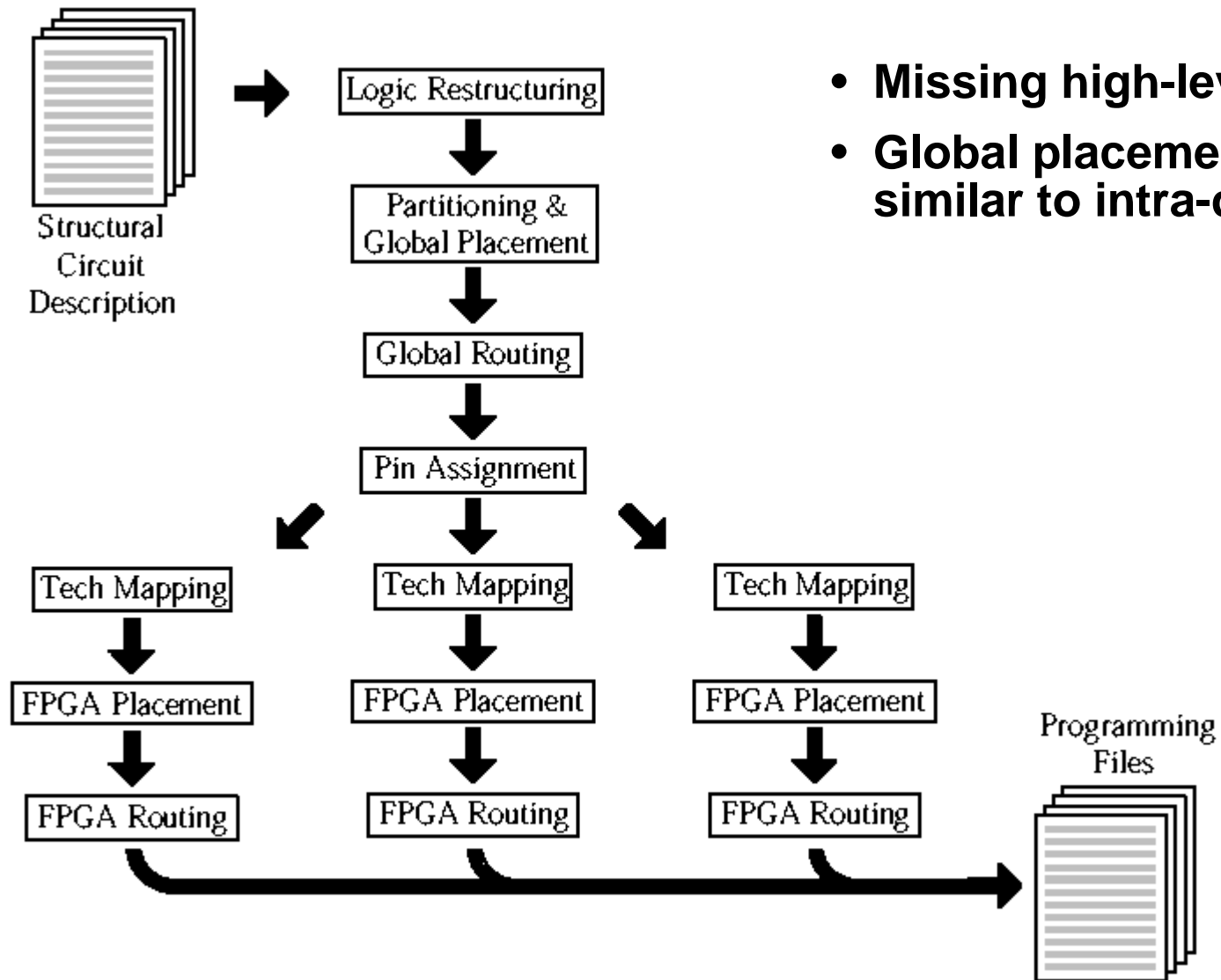
Multi-FPGA System Software



Overview

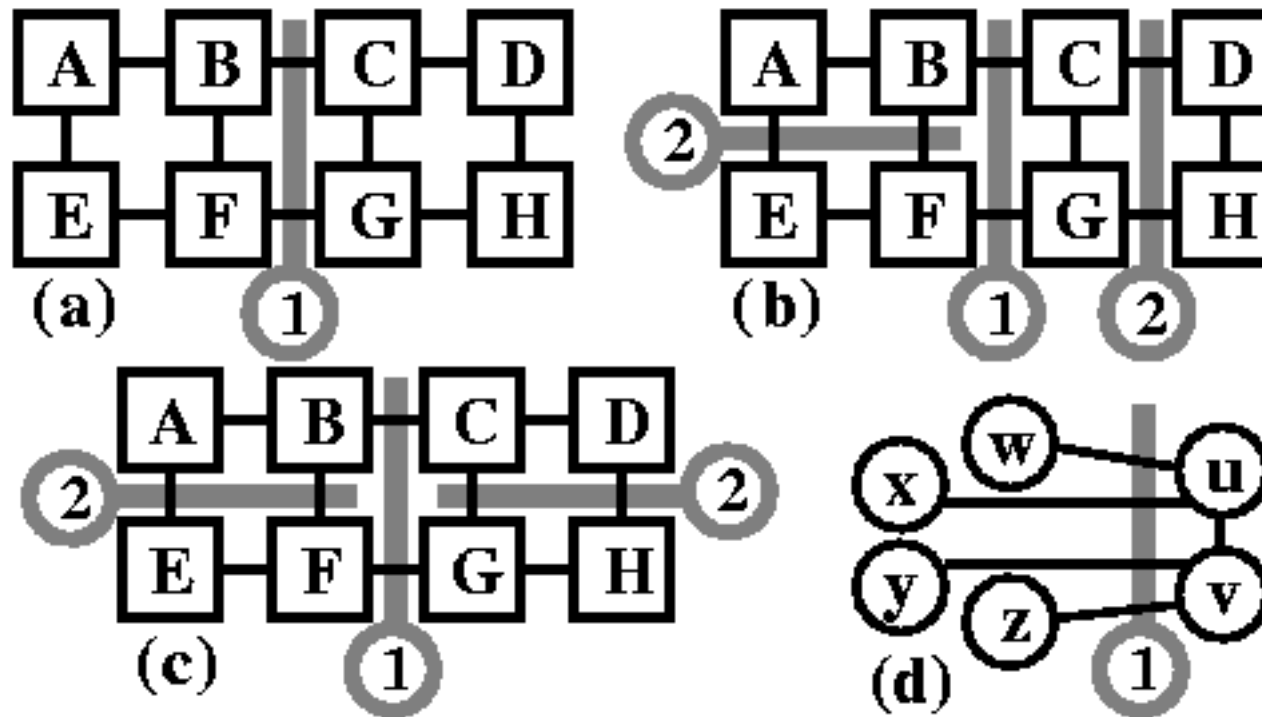
- **Steps in multi-FPGA software**
- **Bipartitioning**
- **Logic Replication**
- **Partition Ordering**
- **Theoretical limits of multi-FPGA systems.**

Multi-FPGA Software



- **Missing high-level synthesis**
- **Global placement and routing similar to intra-device CAD**

System-level Constraints

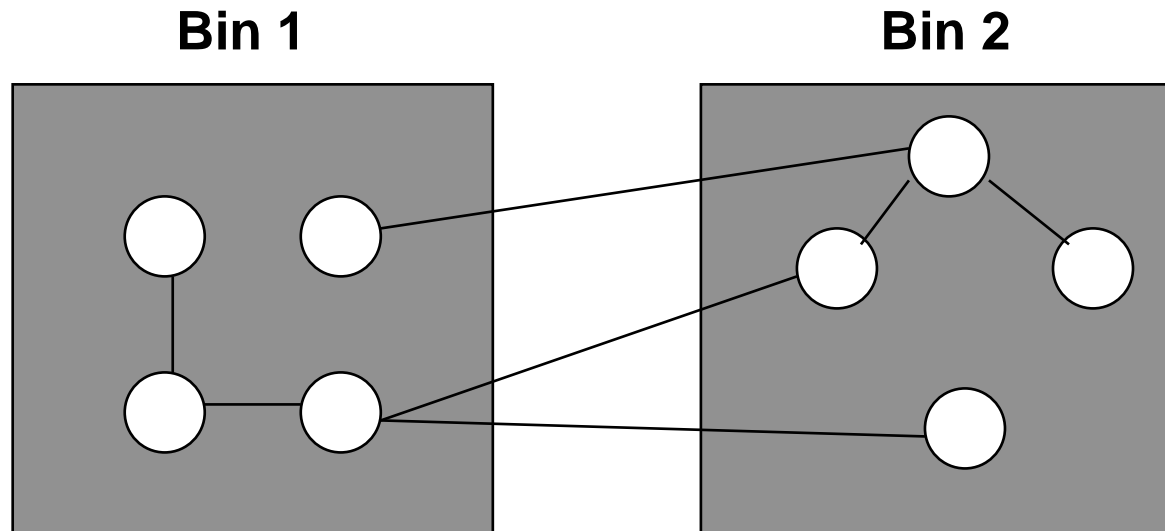


- Even though general solutions are desirable, system specific issues must be considered.
- For many systems, designs are created independently of the system
- Software efficiency determines performance and usability

Bipartitioning

- Perhaps biggest problem in multi-FPGA design is partitioning
- Partitioner must deal with logic and pin constraints.
- Could simultaneously attempt partitioning across all devices. Even “simple” algorithms are $O(n^3)$
- Better to recursively bipartition circuit.

KLFM Partitioning



- Identify nodes to swap to reduce overall cut size
- Lock moved nodes
- Algorithm continues until no un-locked node can be moved without violating size constraints

KLFM Partitioning

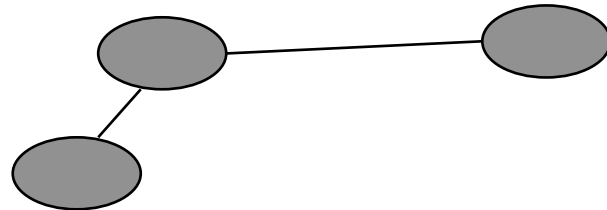
```
Create initial partitioning;
While cutsizes is reduced {
  While valid moves exist {
    Use bucket data structures to find unlocked node in each
      partition that most improves/least degrades cutsizes when
      moved to other partition;
    Move whichever of the two nodes most improves/least degrades
      cutsizes while not exceeding partition size bounds;
    Lock moved node;
    Update nets connected to moved nodes, and nodes connected to
      these nets;
  } endwhile;
  Backtrack to the point with minimum cutsizes in move series just
    completed;
  Unlock all nodes;
} endwhile;
```

Figure 1. The Fiduccia-Mattheyses variant of the Kernighan-Lin algorithm.

- Key issue is implementing node costs in lists that can be easily accessed and updated.
- Many extensions to consider to speed up overall optimization
- Reasonably easy to implement in software

Partition Preprocessing: Clustering

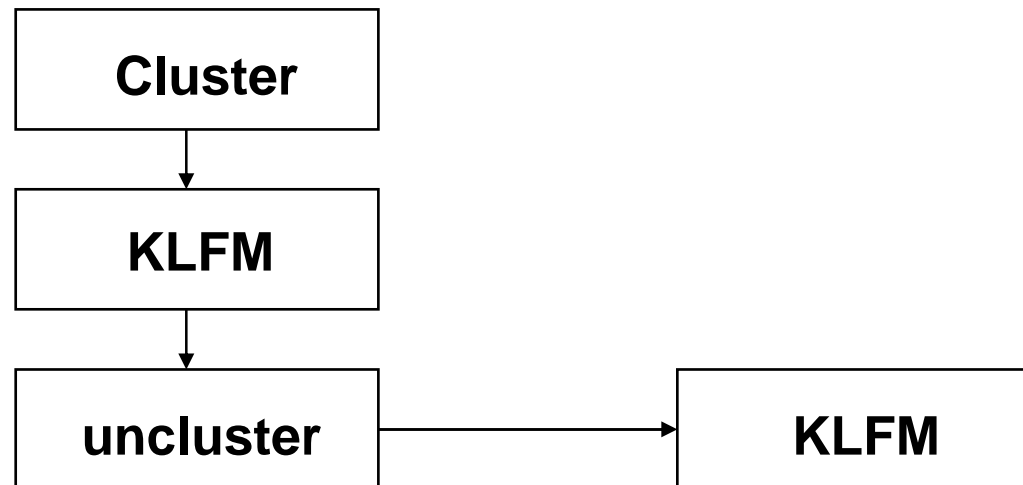
- Identify bin size
- Choose a seed block (node)
- Identify node with highest connectivity to join cluster



- Terminate when cluster size met.
- In practical terms cluster size of 4 works best

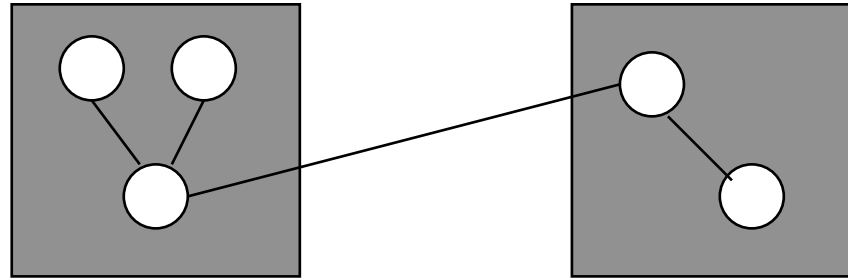
Clustering

- Technology mapping before partitioning is typically ineffective since frequently area is secondary to interconnect
- Frequently bipartitioning continues after unclustering as well.



- This allows for additional fine-grain moves.

Initial Partition Creation



- KLFM primarily designed to operate on fixed-sized partitions.
- Several approaches exist to distribute nodes between the two partitions
 - Random -> assign $\frac{1}{2}$ to each
 - Breadth-first -> select a node, select the next node attached to it
 - Depth-first -> similar to B.F. except get all attached nodes

Partition Creation Results

Mapping	Random	Seeded	Breadth-first	Depth-first
s38417	57	69	57	65
s38584	54	87	56	54
s35932	47	47	47	47
industry3	427	477	427	427
industry2	181	181	181	181
s15850	60	60	60	60
s13207	73	75	80	74
biomed	83	105	104	102
s9234	52	68	52	52
s5378	68	79	80	78
Geom. Mean	82.4	95.3	86.7	86.5
Time	499.5	511.4	498.6	502.1

- Surprisingly random appears to be the best
- For the largest designs, results similar
- For smaller designs, variance across designs
- Seeded ->start from an empty partition and apply KLFM

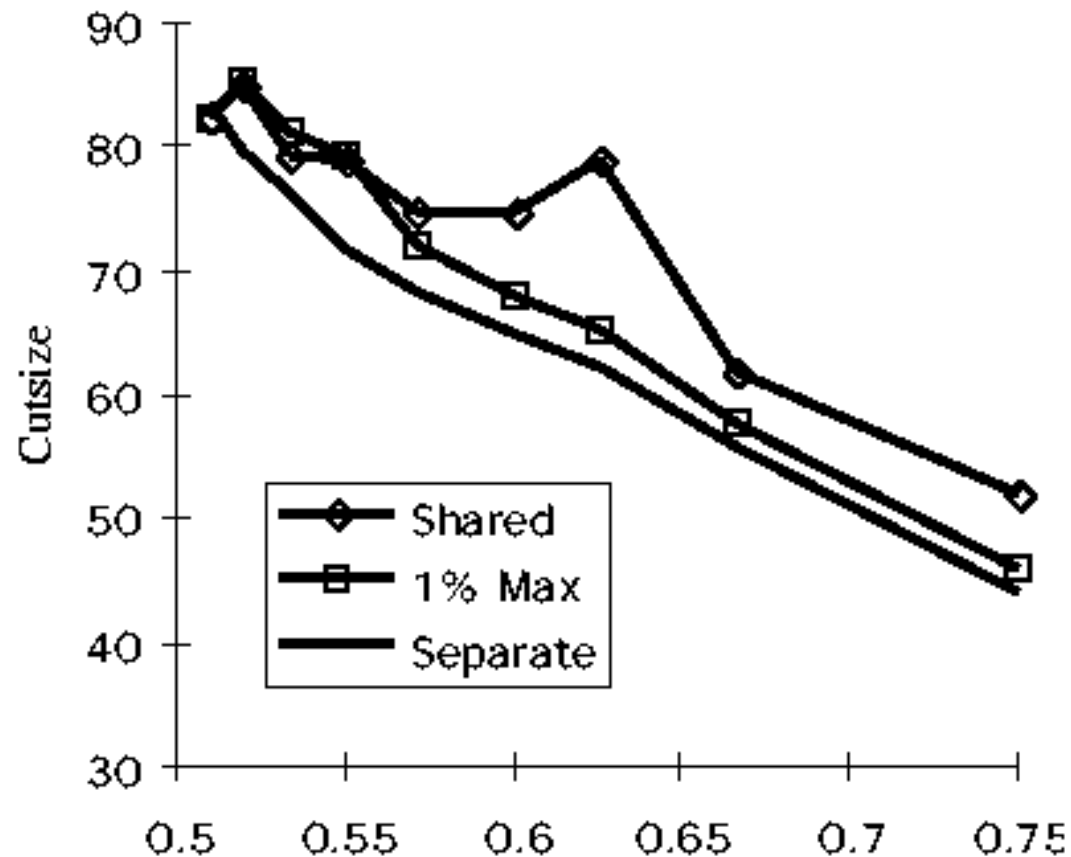
Higher-level Gains



Mapping	Dynamic	1	2	3	4	5	20
s38417	57	56	58	57	57	57	57
s38584	56	57	55	54	54	53	53
s35932	49	47	49	47	47	47	47
industry3	426	426	426	427	427	427	427
industry2	180	208	180	181	173	196	196
s15850	60	64	62	60	60	60	60
s13207	75	77	77	73	73	73	73
biomed	83	98	83	83	83	83	83
s9234	52	56	52	52	52	52	52
s5378	66	71	70	68	68	68	68
Geom. Mean	82.9	87.2	83.9	82.4	82.0	82.9	82.9
Time	532.1	388.6	404.4	499.5	601.4	607.7	936.3

- Effectively look-ahead to try to anticipate next move
- Look-ahead of 3 considered best tradeoff

Partition Size Variation



- Most bipartitions must be balanced so that full FPGA utilization may be achieved
- Frequently application designers do not create circuits that are evenly balanced

Logic Replication

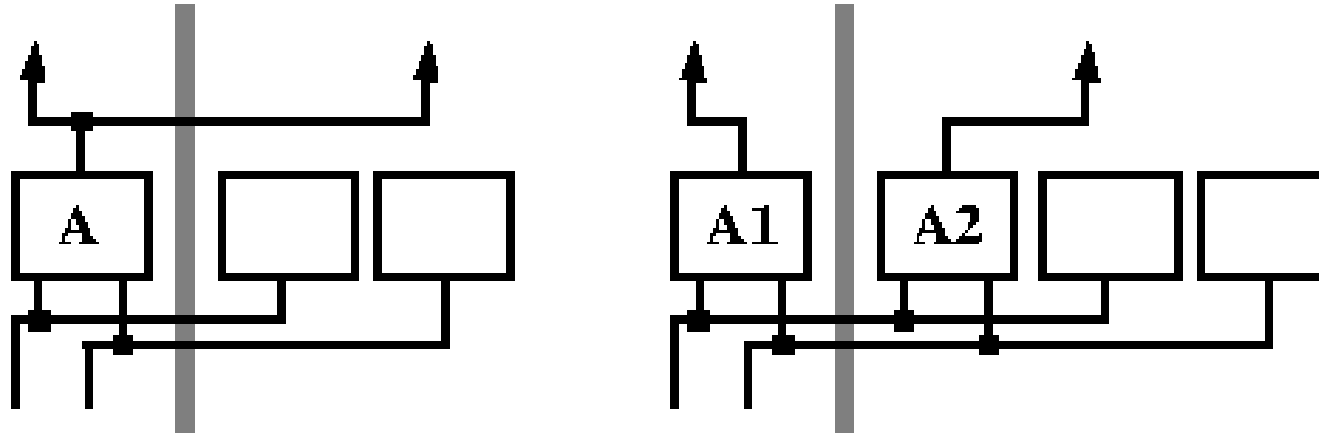
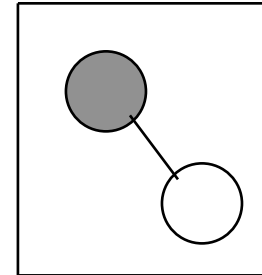
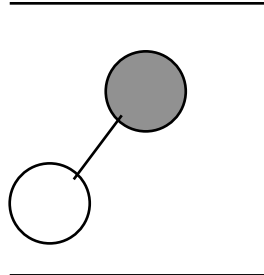


Figure 1. An example of node replication. Node **A** is replicated into **A1** and **A2**, yielding a gain of 1.

- **Attempt to reduce cutset by replicating logic.**
- **Every input of original cell must also input the replicated cell.**
- **Replication can either be integrated into the partitioning process or used as a post-process technique.**

Example: Kring-Newton Replication



- **Introduce a new state to partitioning**
 - Node can exist in separate locations
- **Possible node moves include gain/reduce, replication, and unreplication**
- **Positive unreplication moves must be taken before any other moves**
- **Gradient technique-only allow replication when cut-size changes by more than 10%**

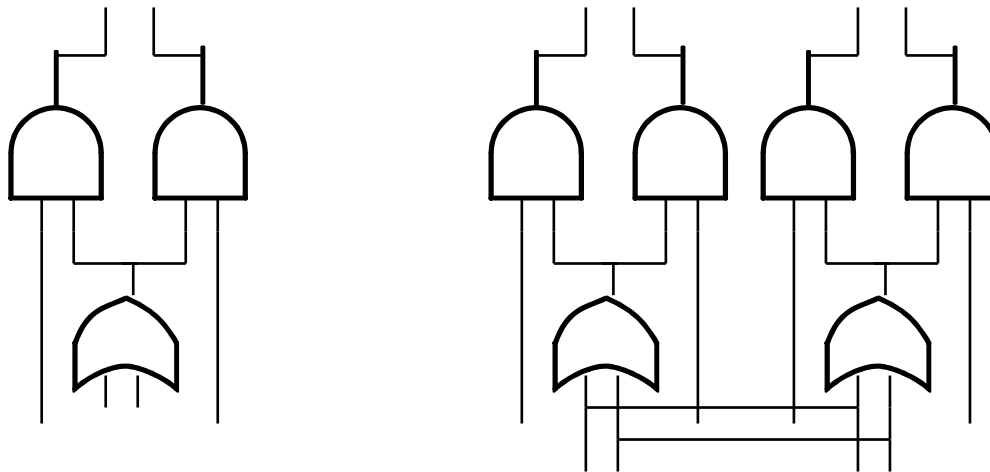
Kring-Newton Results

Bench.	Straw	Basic KN	KN HLG	KN Clust.	KN Grad.
s5378	60	41	39	48	43
s9234	45	30	32	33	32
s13207	67	53	48	49	43
s15850	52	43	43	39	41
C6288	50	33	33	34	33
biomed	161	136	137	126	130
prim2	133	114	114	110	114
struct	33	33	33	33	33
s35932	46	39	39	28	23
s38584	50	35	35	46	32
s38417	55	51	49	47	49
Mean	60.8	48.2	47.7	47.6	44.7
Time	17.4	17.3	17.6	17.1	19.7
Rep %	0.0%	4.2%	4.5%	6.2%	3.5%

- Results indicate 20% improvement in cut size with 5% increase in logic node count.
- Minimal increase in computation time

Functional Replication

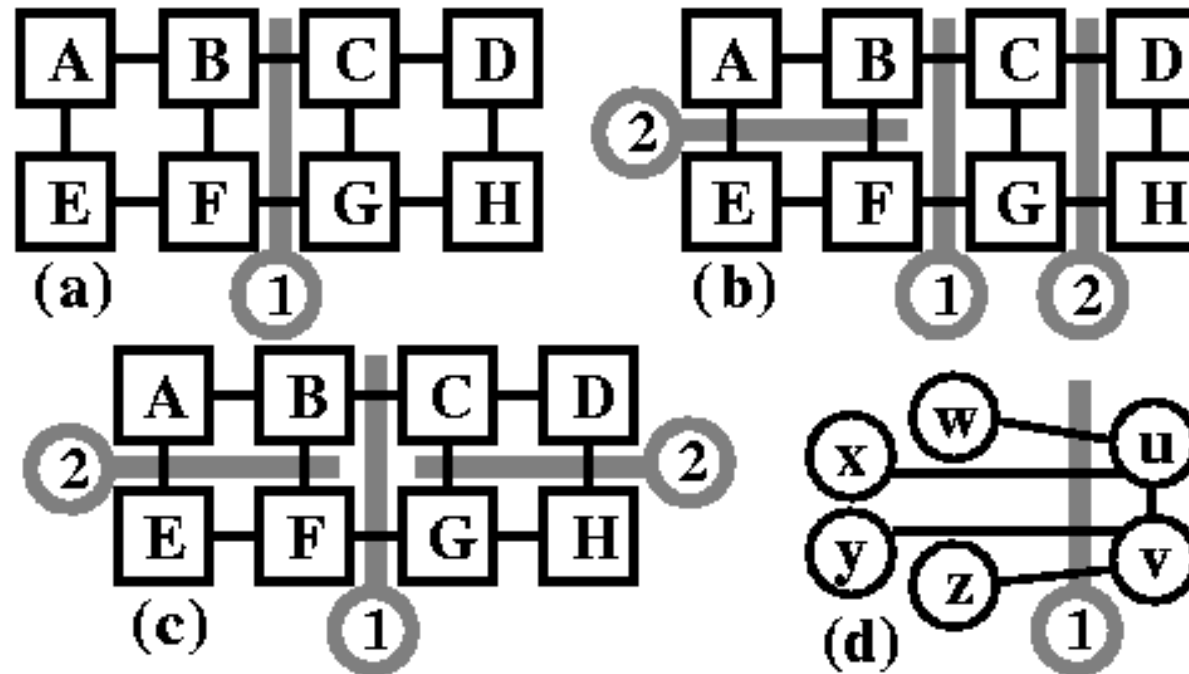
- Applied to tech-mapped Xilinx blocks.
- Outputs in CLBs split into two CLBs
- Only inputs needed by both CLBs split across partitions.



Replication Summary

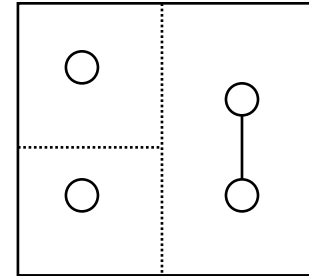
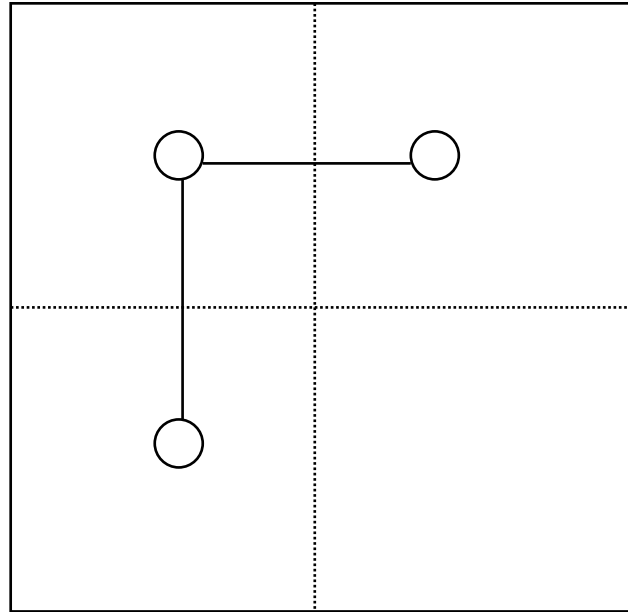
- **Tech mapping before partitioning shown to be ineffective (again)**
- **Kring-Newton simple but effective**
- **Overall summary of bipartitioning**
 - **Use random initial placement**
 - **Bandwidth clustering**
 - **High-order gain of 3 and Kring-Newton to achieve best results**

Logic Partition Ordering



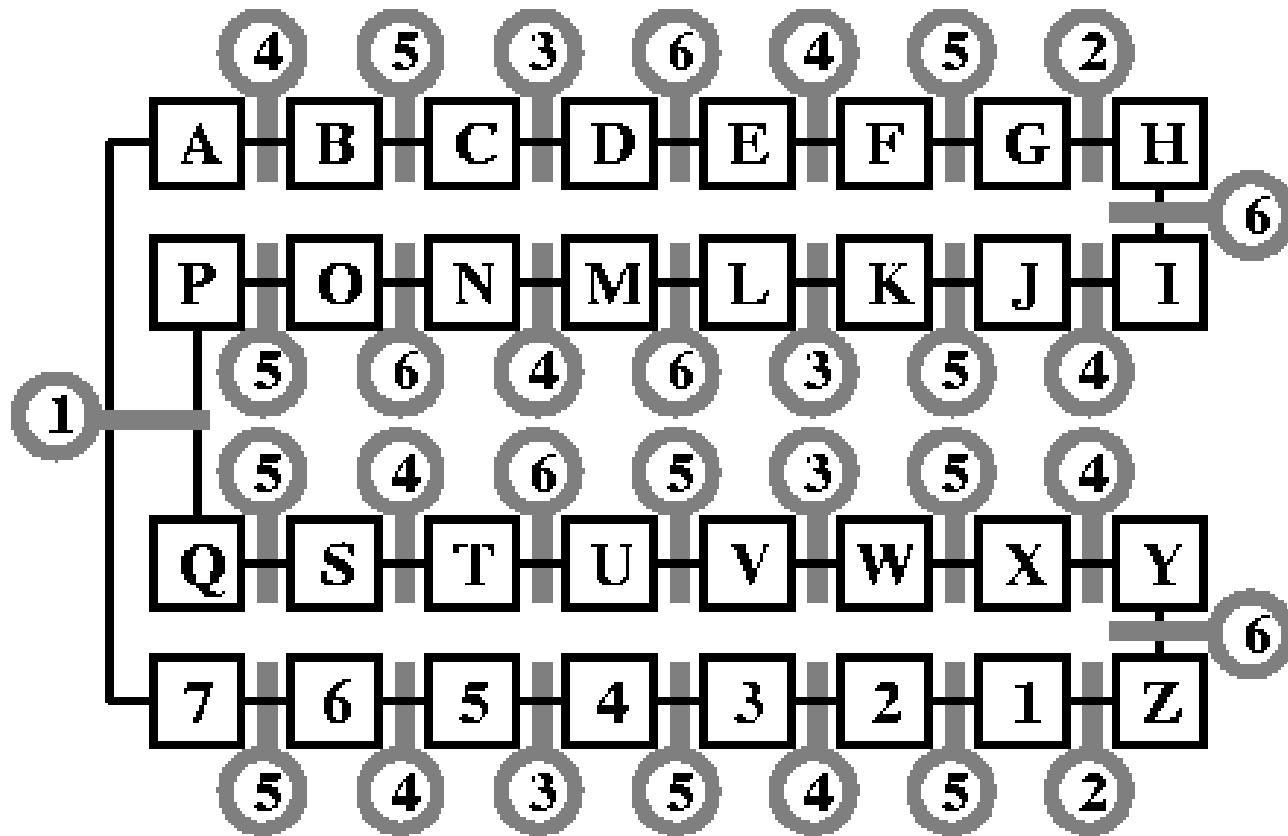
- Simply bipartitioning not enough. Knowing what to partition is important.
- One approach -> locate critical point of expected wires/available wires and partition here first.
- Example above shows alternating horizontal and vertical cuts.

Terminal Propagation



- Even though bipartitioning occurs with a fixed set of nodes, previously cut nodes may play a factor.
- Consider recursive cut. Need to use “anchors” to guide partitioning.

Splash 2



- 68 connections most FPGAs, only 35 between A-7
- More balanced with even schedule
- Somewhat unimportant due to Splash programming style.

Are Meshes Really Realistic?

- The number of wires leaving a partition grows with Rent's Rule

$$P = KG^B$$

- Perimeter grows as $G^{0.5}$ but unfortunately most circuits grow at G^B where $B > 0.5$
- Effectively devices highly pin limited
- What does this mean for meshes?

Summary

- **Multi-FPGA system software requires many steps.**
- **Bipartitioning has been the subject of much research**
- **Surprisingly, simple approaches to initializing partitions and replicating logic is most effective.**
- **Pin limitations pose a problem -> address this issue in next class.**