CAD Problems for Multi-FPGA System

Multi-FPGA System

- A multi-FPGA system is made up of a set of interconnected FPGAs.
- A large circuit needs to be partitioned into subcircuits that satisfy the I/O constraints and the capacity constraints of the FPGAs in a multi-FPGA system (e.g. commercial emulation systems).
- Each subcircuit is placed and routed on a single FPGA.
- In addition, inter-FPGA connections between the FPGAs also have to be routed.

Partitioning

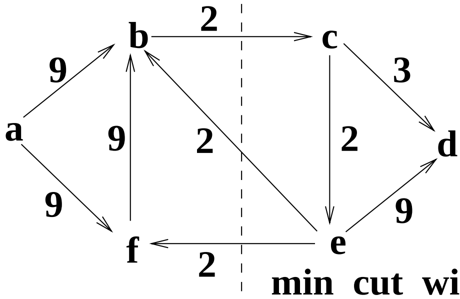
- In multiple-FPGA partitioning, a large circuit is partitioned between multiple FPGA chips so that each subcircuit can be fit into a single FPGA.
- FPGA partitioning problems are constraint-driven. A partitioning solution must satisfy the I/O constraint as well as the capacity constraint of each FPGA.
- FPGA partitioning algorithms implemented in commercial CAD tools are mostly move-based approaches like Fiduccia-Mattheyses algorithm with some modification to take care of FPGA-specific constraints.
- FPGA device is becoming more heterogenous in terms of the types of resources (e.g. wide input gates, SRAM arrays for on-chip memory, different types of logic blocks), so the capacity constraints for a FPGA can be very complex.

Partitioning with Logic Replication

- In multiple-FPGA partitioning, the objective is to minimize the number of FPGAs required subject to I/O constraints and logic capacity constraints of the FPGAs.
- FPGAs tend to be more pin-limited than logic-limited.
- Selectively replicating some logic blocks of a circuit to different FPGAs can reduce the number of inter-FPGA connections.
- When applied carefully logic replication can reduce the number of FPGAs required.

Classical Bipartitioning

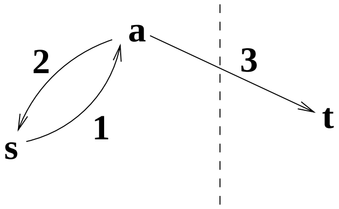
- Model the given circuit as a graph.
- Compute a cut with a small size to divide it into two disjoint components.



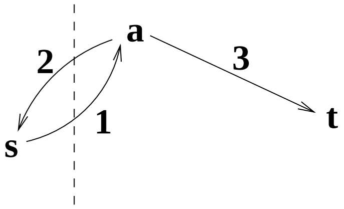
min cut with size = 6

Replication Bipartitioning

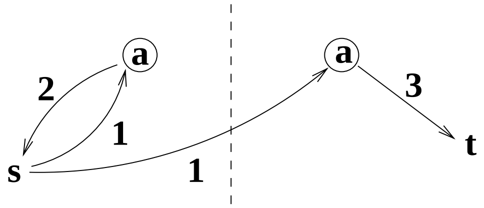
- Allow some nodes to appear on both components.
- Node replication can improve cut size.



Node a on the left, cut size = 3



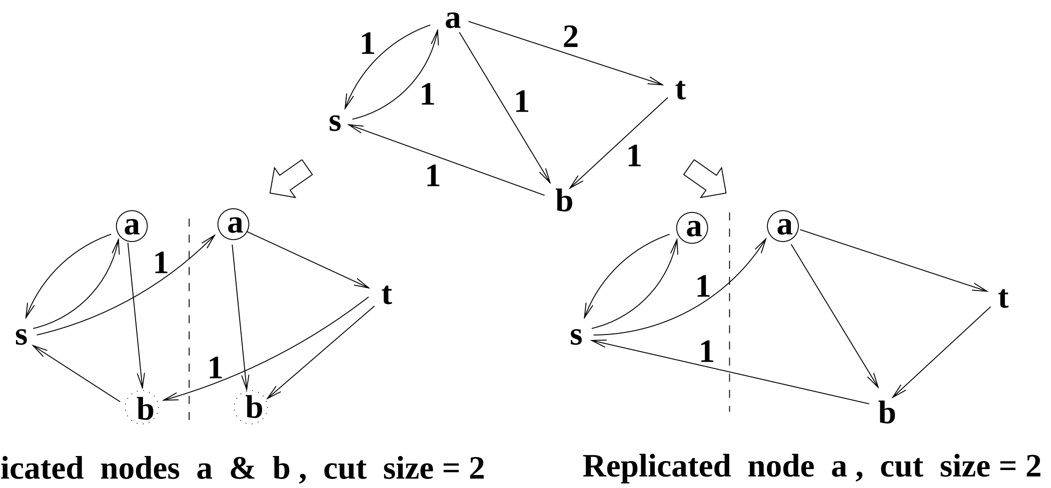
Node a on the right, cut size = 3



Node a replicated, cut size = 1

Replication Min-Cut Partitioning

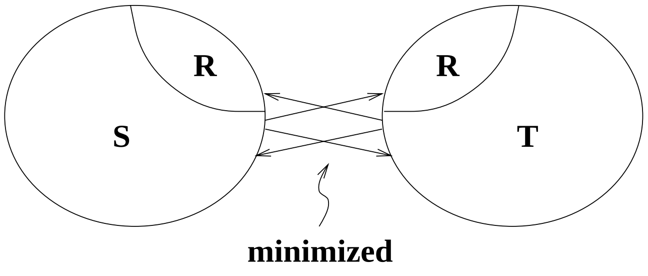
- Compute a partition of the minimum cut size with node replication allowed.


a
2
1
1
1
t
s
1
1
b
a
a
1
t
s
1
b
b
a
a
1
t
s
1
b
Replicated nodes a & b , cut size = 2
Replicated node a , cut size = 2

- If the amount of replication used to achieve the minimum cut size is minimized, it is called a *Minimum Replication Min-Cut Partitioning*.

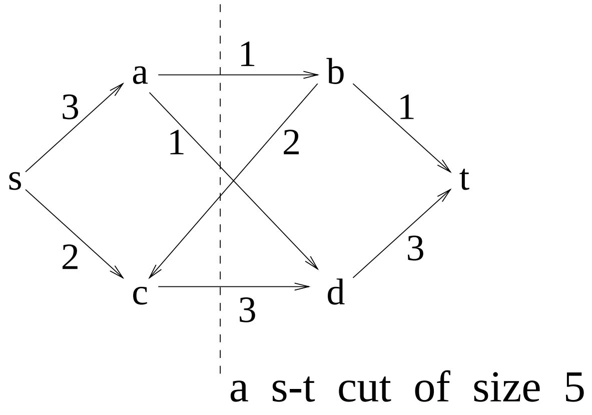
Minimum Replication Min-Cut Partitioning

- Given two nodes s and t to be separated.
- Partition node set V into S, T, R with $s \in S$, $t \in T$.
- Minimize size of cut $(S \cup R, R \cup T)$ using the smallest possible R. (Note that R is the set of replicated nodes.)

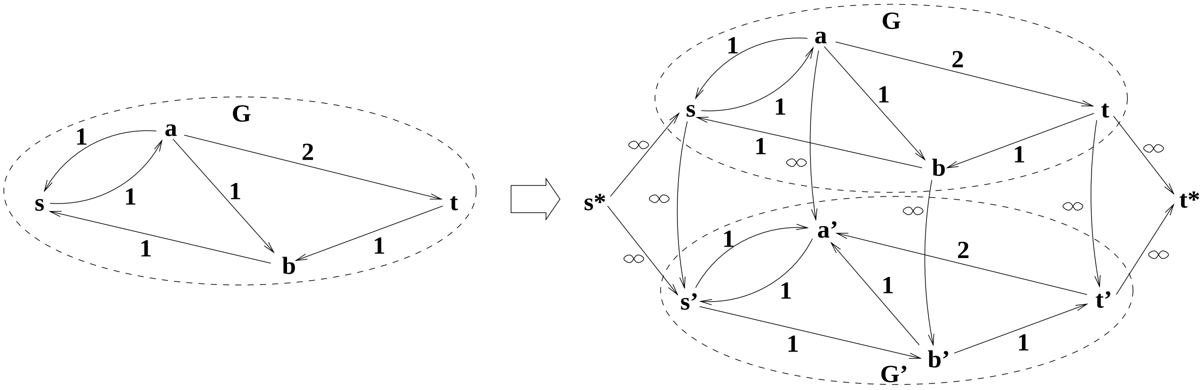


Definition: A s-t *cut* of a directed graph is a cut that separates nodes s and t.

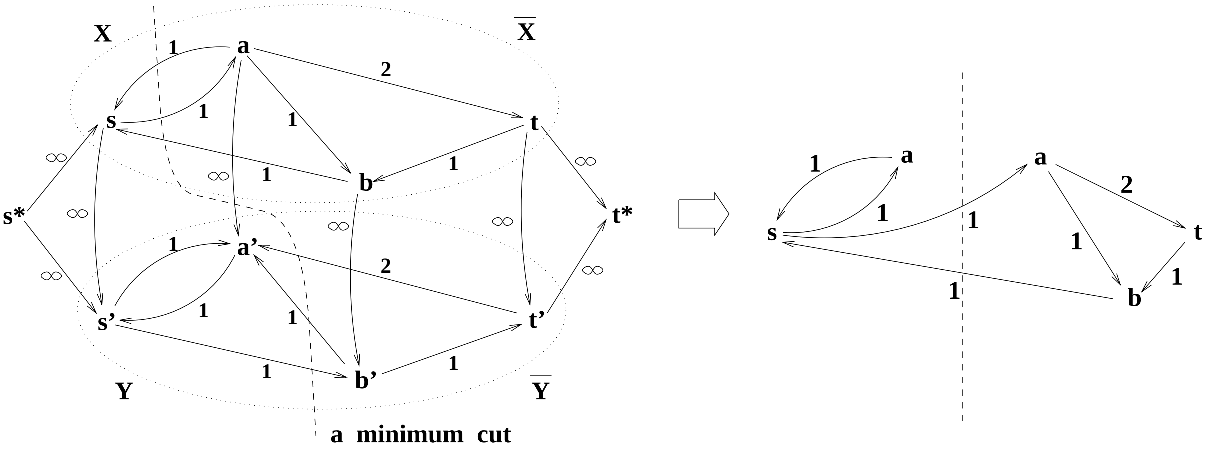
Definition: The size of a s-t cut is the sum of the weight of all edges directed from the side of s to the side of t.



- Note that we want to count edges running from one side to another and vice versa.
- A *replication network* which consists of a copy of the original graph G and a mirror image G' of the graph will be useful.

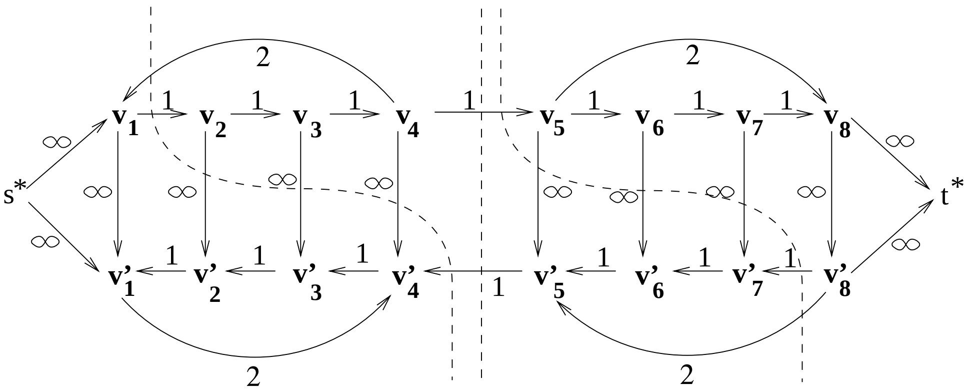


Theorem: Any s-t minimum cut of the replication network defines a replication min-cut partition $(S \cup R, R \cup T)$ where $S = X$, $T = \{u : u' \in \bar{Y}\}$, $R = V - S - T$ [Liu, Kuo, Cheng, Hu '95].



$S = \{s\}$, $T = \{b, t\}$, $R = \{a\}$.

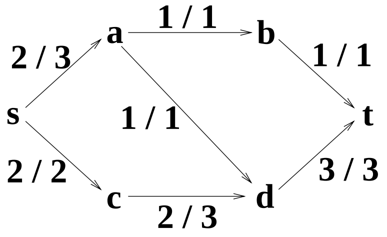
But there are possibly many s-t minimum cuts $(Z, \bar{Z})$ in a replication network. How to find one that requires the least replication?



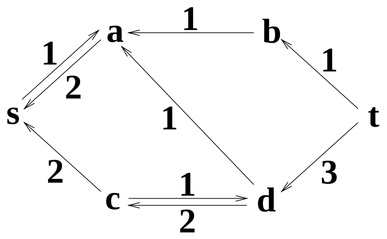
Note: A node u is replicated iff $u' \in Z$ and $u \in \bar{Z}$.

Network Flow

Definition: Given a directed graph with capacitiated edges, a maximum flow is a way to ship the maximum possible amount of flow from source node s to sink node t.



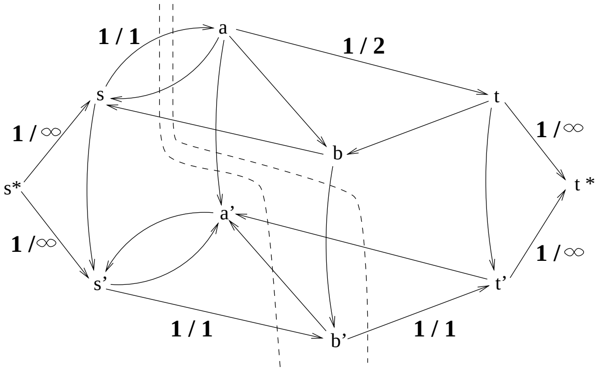
The *residual network* w.r.t. a flow has edge (u, v) s.t. $c(u, v) =$ additional amount of flow that can be pushed from u to v.



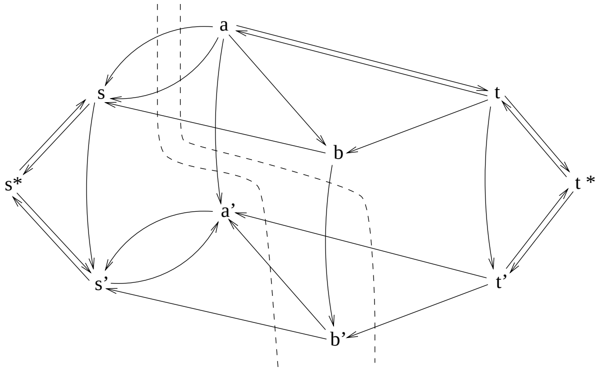
Minimum Replication Min-Cut Partitioning Algorithm

1. Construct residual network w.r.t. a max flow in the replication network.
2. Find a s-t min cut in residual network with penalty added for each node u included in replication set R.

Details
1. Construct residual network w.r.t. a maximum flow



A maximum flow

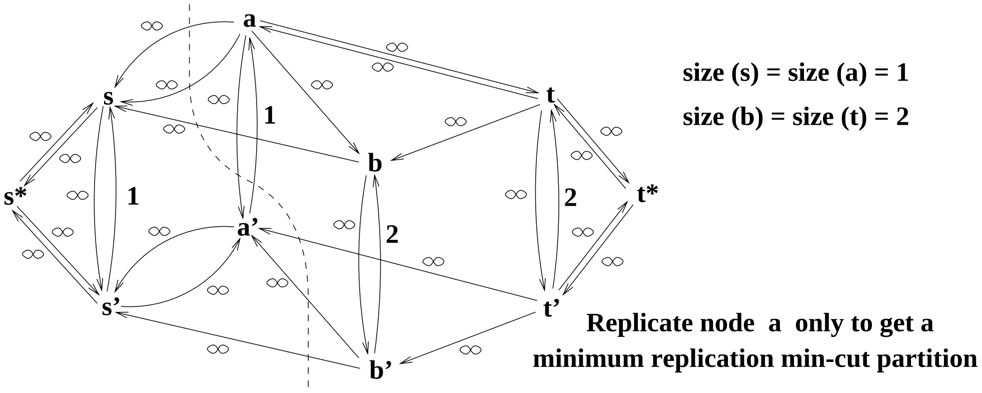


Residual Network

Lemma: The s-t min cuts in the residual network are exactly those in the original network.

2. Find a s-t min cut in residual network with penalty added for each node u included in replication set R

- set cost of all arcs in residual network to ∞
- add $arc(u', u)$ to residual network for all node u
- set cost of $arc(u', u)$ = size of node u



size (s) = size (a) = 1

size (b) = size (t) = 2

Replicate node a only to get a minimum replication min-cut partition

Theorem: Any s-t min cut in the resultant network defines a minimum replication min-cut partition [Mak & Wong '96].

Experimental Results

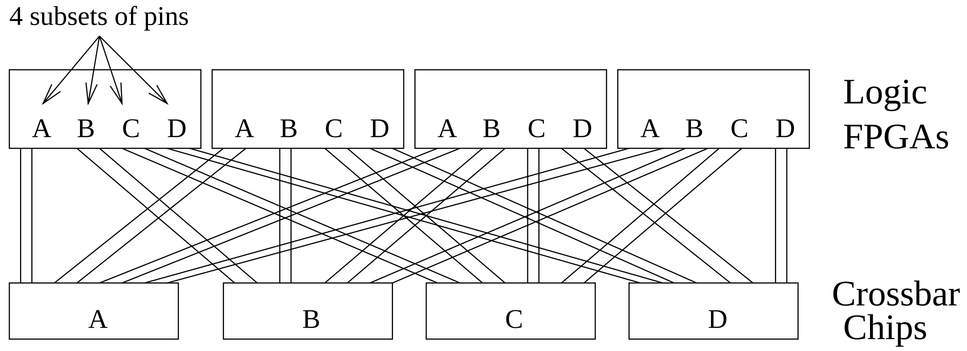
Circuit	Replication Size	
	By arbitrary s-t min cut in replication network	Ours
C880	28.72%	14.36%
C1355	21.61%	5.31%
C1908	30.23%	13.52%
C2670	26.15%	9.30%
C3540	33.85%	11.50%
C5315	27.61%	7.89%
C6288	24.59%	11.38%
C7552	31.35%	11.13%
s1196	32.78%	12.09%
s1494	28.18%	7.35%
s5378	28.37%	11.48%
s9234	25.90%	9.00%
mean	28.28%	10.36%

Inter-FPGA Routing Architectures

- The inter-FPGA routing architectures used by different multi-FPGA systems can be boardly classified into two types:
  - Direct connection architecture
  - Indirect connection architecture using specialized interconnection chips
- With the direct connection architecture, the FPGAs are hardwired together in some pre-defined manner.
- With the indirect connection architecture, the FPGAs are not hardwired to each other but are hardwired to some programmable interconnection chips.
- The second scheme is usually preferred since it is more flexible. Moreover, the first scheme will take up a portion of the routing resources within each FPGA for inter-FPGA connections which means more FPGAs have to be used.

Partial Crossbar Inter-FPGA Routing Architecture

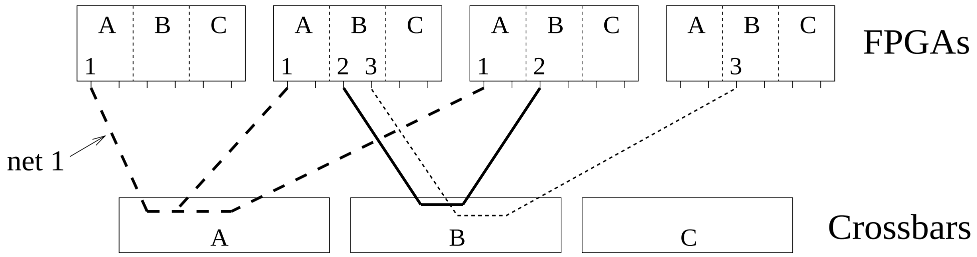
- Considered one of the best inter-FPGA routing architectures [Kim & Shin '96].
- Have been used in commerical logic emulation systems e.g. Quickturn.
- I/O pins of each FPGA are divided into groups.
- I/O pins of the same group from different FPGAs are connected to the same crossbar.
- I/O pins of the same group but from different FPGAs can be connected by programming the switches inside a crossbar.
- Since the number of crosspoints required to form a crossbar grows as the square of the pin-count of the crossbar, a number of small crossbars with smaller pin-count each are used instead of a single crossbar with a large pin-count.



A multi-FPGA system using the partial crossbar inter-FPGA routing architecture.

Routing Through Crossbars

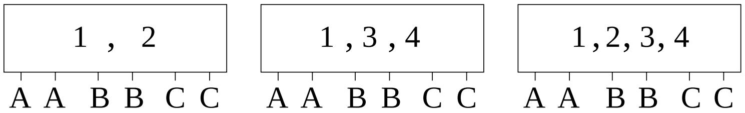
- **Goal:** Connect all inter-FPGA nets through the crossbars.
- **Fact:** Terminals of the same net can be connected via crossbar α when they are connected to I/O-pin subset α of the FPGAs by turning on the appropriate switches in crossbar α.



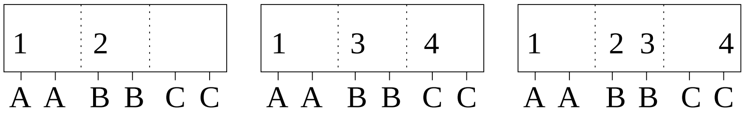
Problem Formulation

- Assign each inter-FPGA net to a pin-subset so that no more than P nets are assigned to the same pin-subset in any FPGA where P is the # of I/O pins in a pin-subset.

A routing instance with P = 2 :

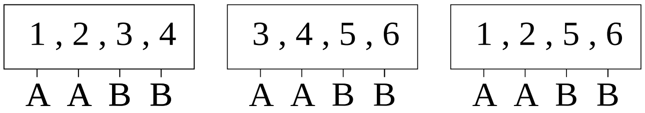


A feasible assignment of nets to pin-subsets :

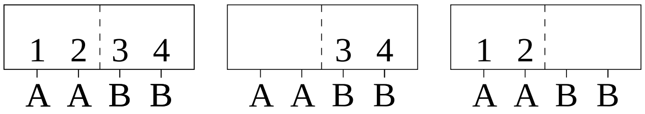


- A feasible assignment also corresponds to a feasible routing solution of all nets.

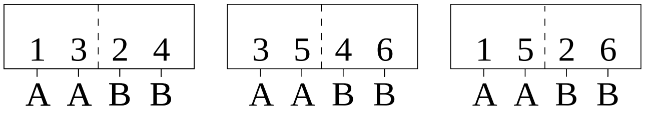
- Heuristics have been proposed to solve the problem.
  E.g. Process the nets one by one. Assign each net to the first subset that still has pins available on all the FPGAs that contain a terminal of the net.



Heuristic can get stuck without finishing routing all nets:



Even though feasible routing solution does exist:



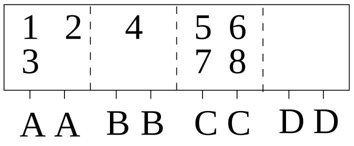
- **Theorem:** If even pin subset size is used, any routing instance for which all nets are 2-terminal nets is always routable (i.e., a feasible routing assignment can be found). [Mak & Wong ’95a]
- An efficient algorithm based on Euler circuit computation can be used to compute a feasible routing assignment solution in the case above.

Algorithm for 2-Terminal Net Routing

Notation:
$T(v) =$ no. of nets assigned to pin subset T in chip v

$\Phi(v) =$ total *overflow* at chip v
$= \sum_{T:T(v)>P}(T(v) - P)$ where $P =$ # pins in a subset.

e.g. v :



$\Phi(v) = (A(v)-2) + (C(v)-2)$
$= 3$

Note: An assignment is feasible $\Longleftrightarrow \Phi(v) = 0$ for all chip v

Idea

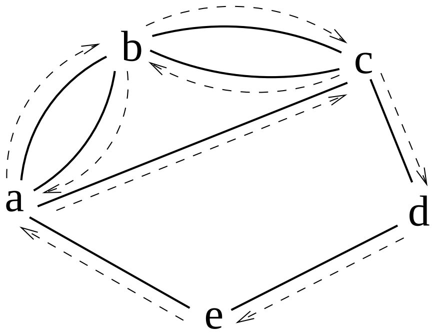
- Start with an arbitrary (infeasible) assignment.
- Reduce total overflow at each chip iteratively until it becomes 0 without increasing the overflow at other chips.

Algorithm

Repeatedly do the following until no chip has an overflow:

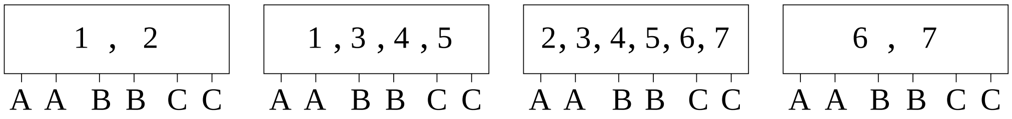
1. If $\Phi(v) > 0$ for chip v, then find subsets T and T' s.t. $T(v) > P$ and $T'(v) < P$
2. Balance the no. of nets assigned to subsets T and T'
   - by re-assignment
   - based on finding an Euler circuit in a graph

Definition: An Euler circuit of a graph G is a closed path that traverse every edge in G exactly once.

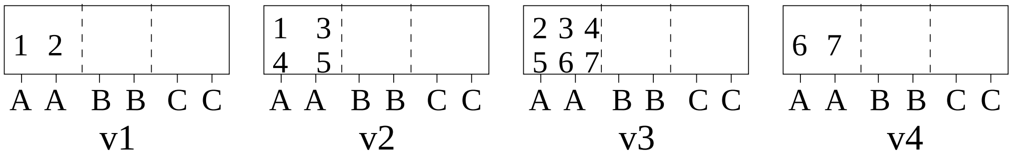


Theorem: A graph has an Euler circuit iff it is connected and all its vertices have even degrees.

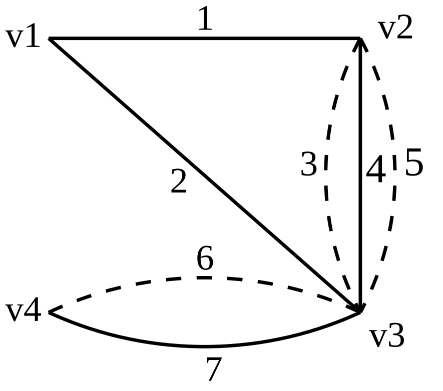
An instance with *P* = 2



0. Initial assignment :



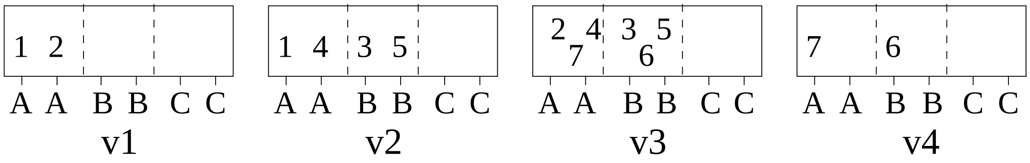
1. A(v2) > *P* > B(v2)



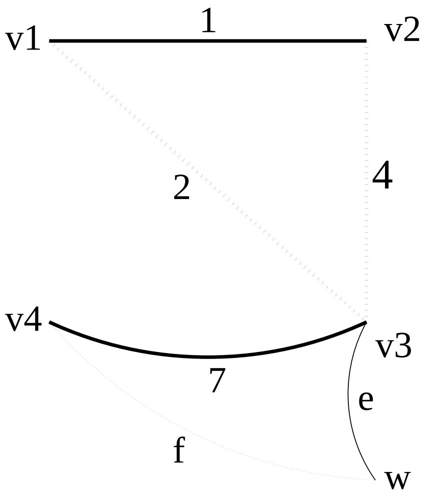
Euler circuit : 1-3-4-5-7-6-2

——— : re-assign to A

- - - - : re-assign to B



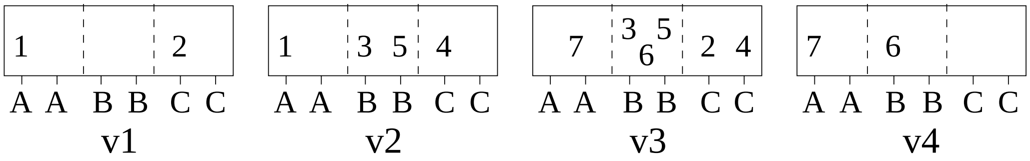
2. A(v3) > *P* > C(v3)



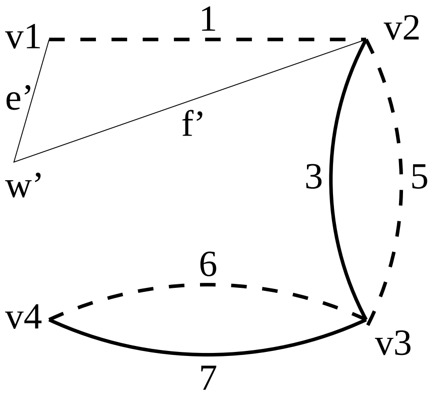
Euler circuit : e-4-1-2-7-f

——— : re-assign to A

·········· : re-assign to C



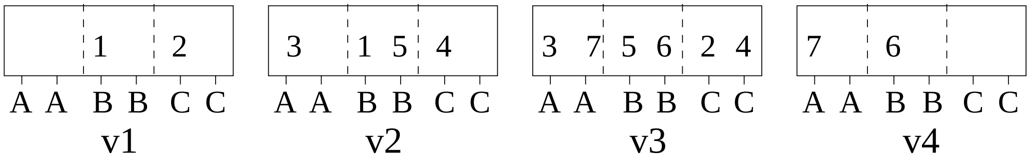
3. B(v3) > *P* > A(v3)



Euler circuit : e’-1-3-6-7-5-f’

——— : re-assign to A

– – – – : re-assign to B



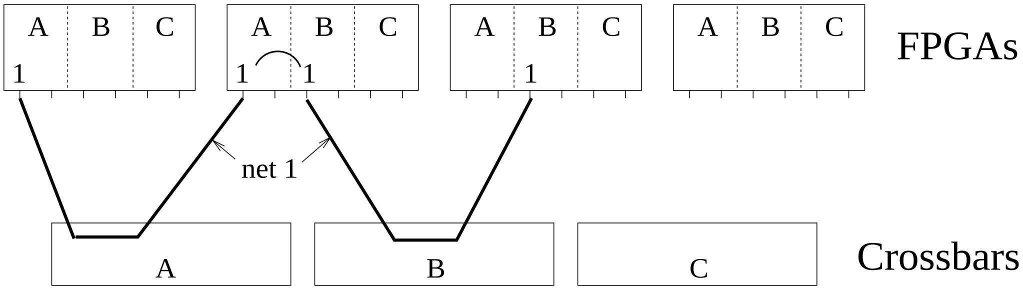
Multi-Terminal Net Routing

Theorem: The board-level routing problem with multi-terminal nets is NP-complete [Mak & Wong ’95a].

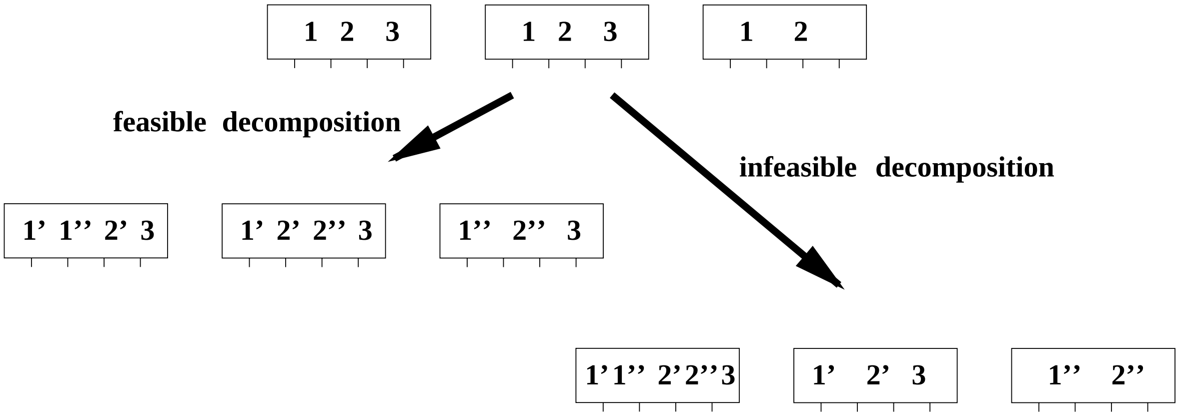
Proof: Reduction from Graph K-Colorability (K = no. of subsets per chip) .

Solution for Multi-Terminal Net Routing

- Decompose multi-terminal nets into 2-terminal nets and allow some terminals to connect to more than one I/O-pin:



- Then any two-terminal net routing algorithm can be used to route the decomposed nets.



An algorithm to simultaneously decompose all multi-terminal nets subject to the I/O pin limit on each FPGA whenever a feasible decomposition exists was developed [Mak & Wong '95b].

Lemma: All nets will be successfully decomposed if # of I/O-pins on each FPGA is no less than $\sum_{i\geq 2}\left\lceil\frac{(2i-2)\triangle_i}{i}\right\rceil$ where $\triangle_i$ is the maximum no. of i-terminal nets in a FPGA.

Remark: Proportion of multi-terminal nets is usually small. e.g. If proportion of 3-terminal nets in each FPGA is $< 15\%$, proportion of 4-terminal nets in each FPGA is $< 5\%$, and the rest are 2-terminal nets, then we need less than 7% of extra I/O pins.