
ECE 636

Reconfigurable Computing

Lecture 4

FPGA Placement



Outline

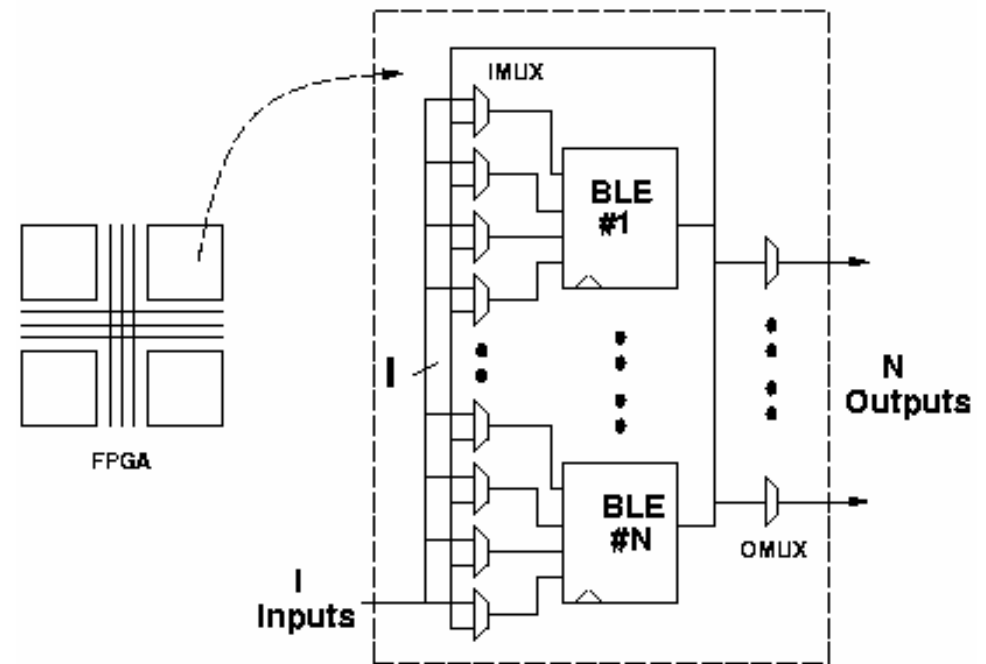
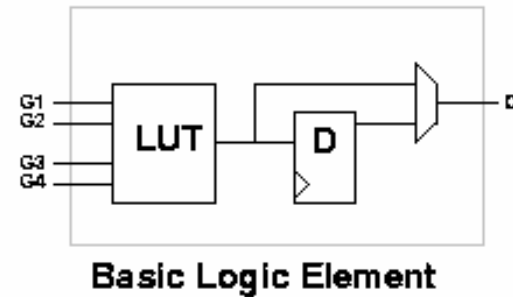
- **Brief review of important architecture info for homework**
- **Basic clustering of BLEs into clusters**
- **Timing-driven analysis and clustering**
- **Placement techniques (simulated annealing)**
- **Timing driven placement**

Placement metrics

- **Quality metrics for layout:**
 - Area
 - Delay
 - Dynamic power consumption (relatively recently)
- **Ideally placement and routing would be performed together**
 - Some have tried!
 - Both problems are NP-hard
 - For practical considerations placement and routing must be performed separately

Before Placement: Clustering

- Need to group BLEs into groups
- Goals:
 - Minimize number of clusters
 - Minimize inter-cluster wiring
 - Minimize critical path (timing-driven)
- How do we do this
 - Take advantage of cluster architecture



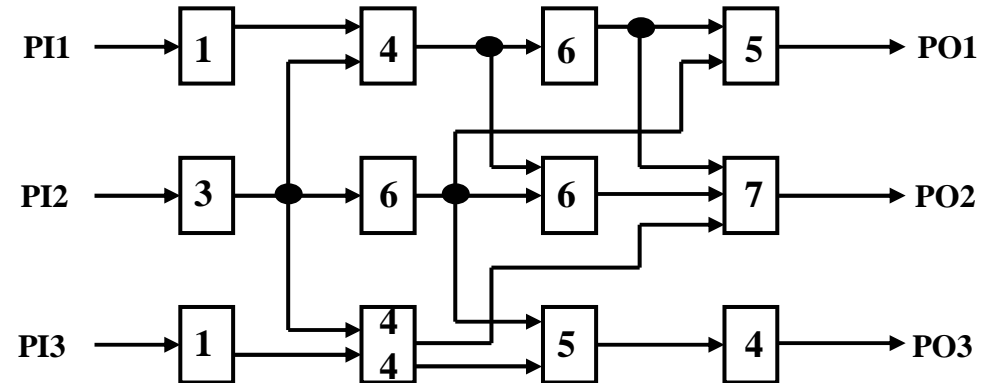
Basic Clustering (Betz) – CICC 97

- **Iterate until all BLEs consumed**
 - **Start new cluster by selecting a random BLE**
 - **Add BLE with most shared inputs with current cluster to cluster**
 - **Keep adding until either cluster full or input pins used up**
 - ***Hill climbing – if some cluster BLEs unused***
 - **Add another BLE even if cluster input count temporarily overflowed**
 - **If input count not eventually reduced select best choice from before hill climbing**

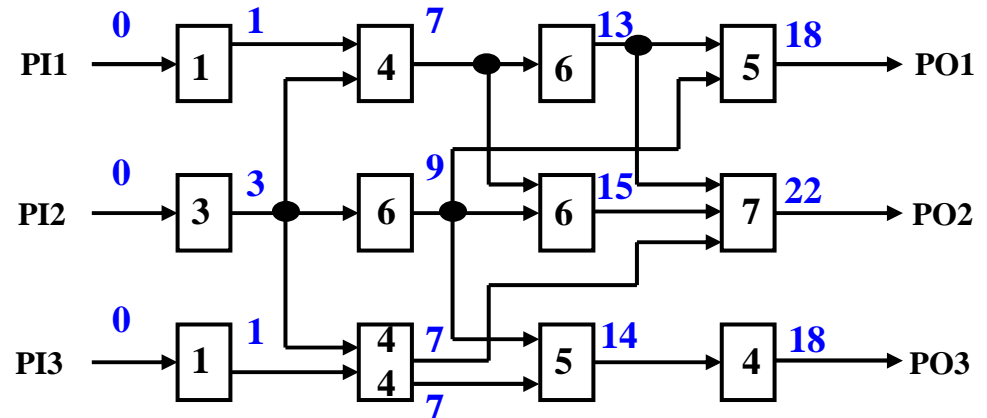
Does this do anything for timing performance?

Timing Analysis

netlist with delay for each gate



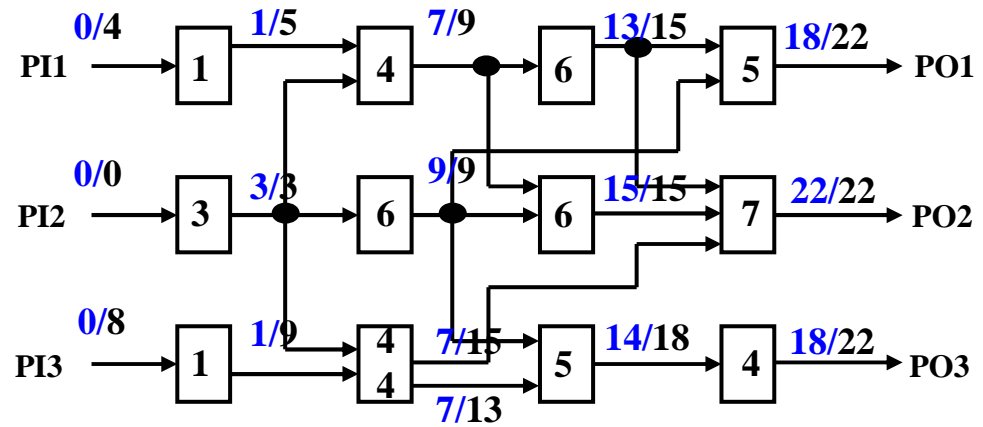
arrival times



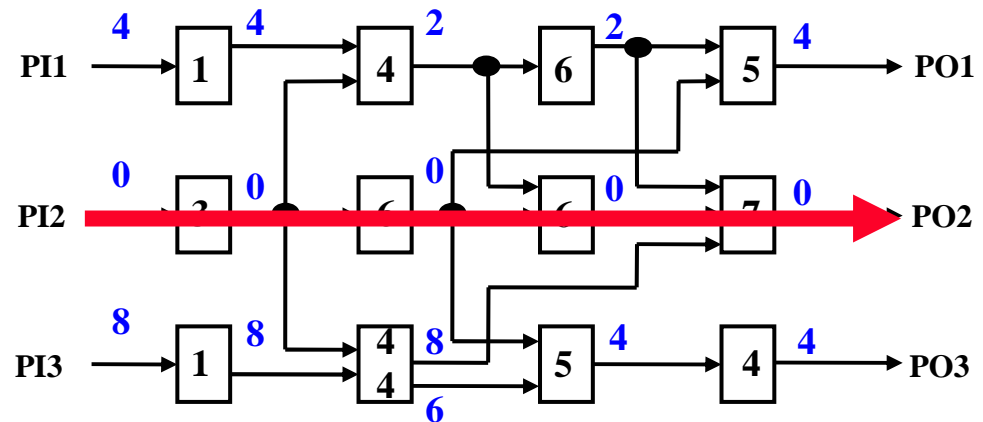
Source: David Pan

Timing Analysis

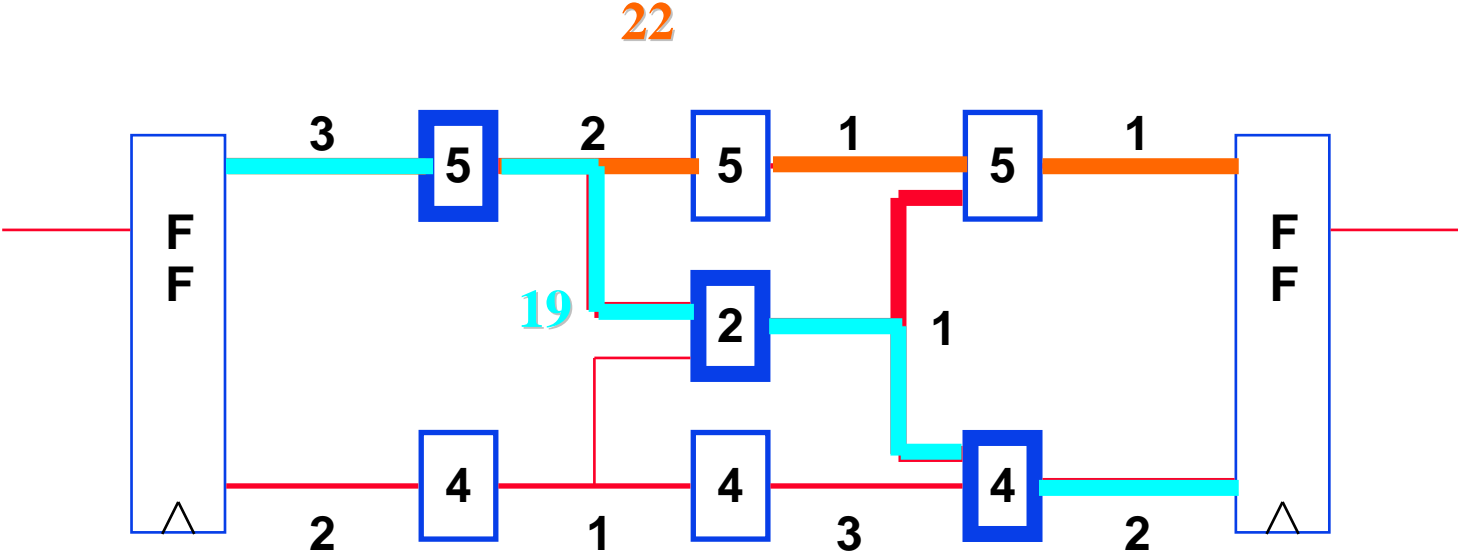
arrival time/required time



slack = required time - arrival time



Example with interconnect delay



Timing-Driven Clustering – T-VPACK

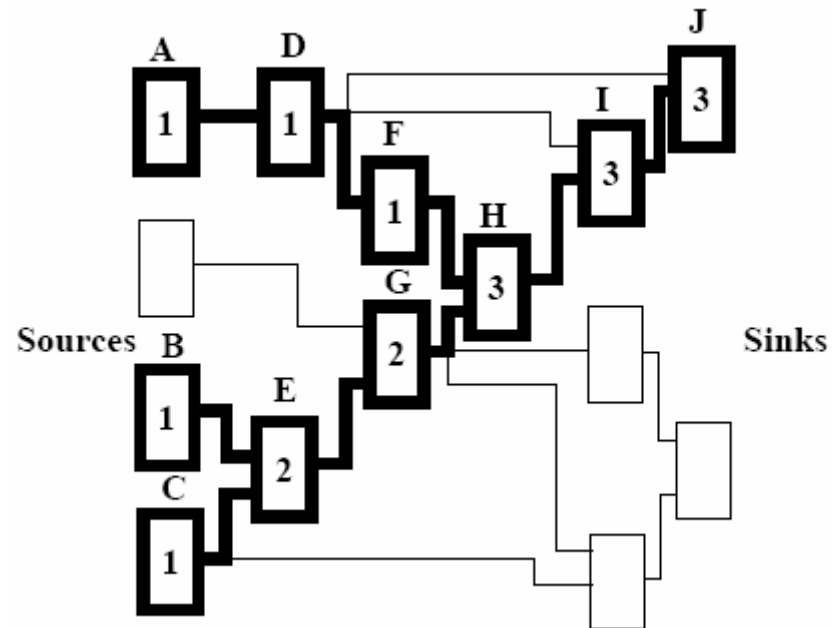
- **Cost metric now considers both connectivity and timing criticality**

$$Connection_Criticality(i) = 1 - \frac{slack(i)}{MaxSlack}$$

- **Perform an analysis of criticality at beginning considering all wires to be inter-cluster**
- **As clustering progresses consider the following timing weight ratios**
 - LUT delay: 0.1
 - Intra-cluster delay
 - Inter-cluster delay
- **Determine “Base” BLE criticality**

How to break ties?

- Initially, many paths may have the same number of BLEs



- Include “tie-breaking” in performance cost function

$$Criticality(B) = Base_BLE_Criticality(B) + (\epsilon \cdot total_paths_affected(B))$$

$$Attraction(B) = \alpha \cdot Criticality(B) + (1 - \alpha) \cdot \frac{|Nets(B) \cap Nets(C)|}{G}$$

Results for T-VPACK versus VPACK

- Timing driven place and route also used for these results.

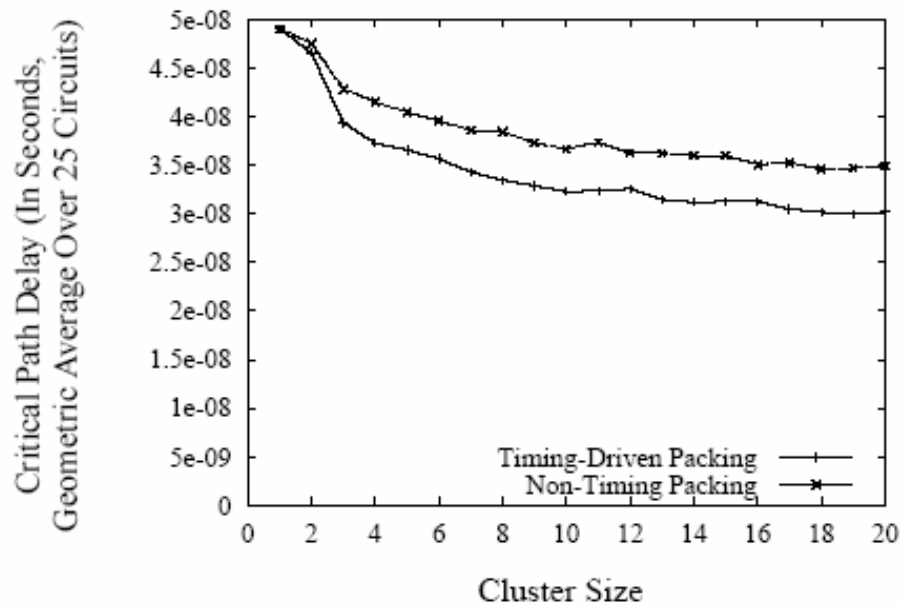


Figure 9. Critical Path Delay vs. Cluster Size

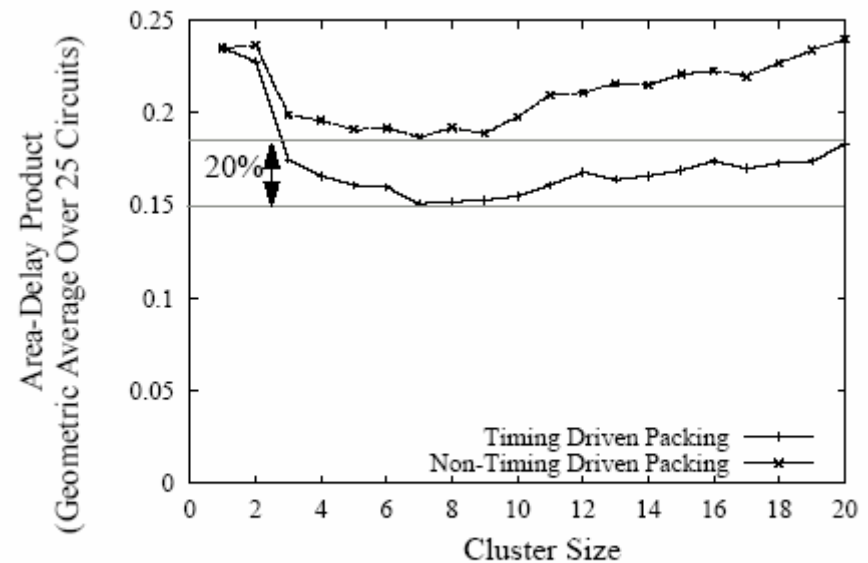
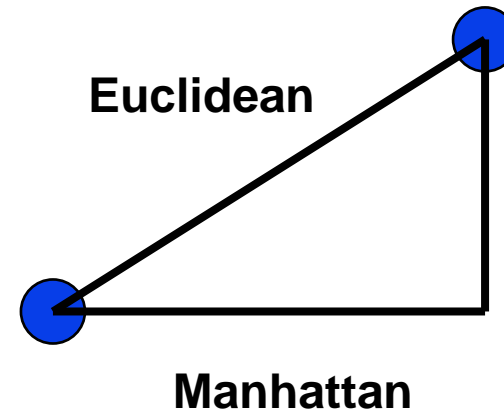


Figure 10. Area-Delay Product vs. Cluster Size

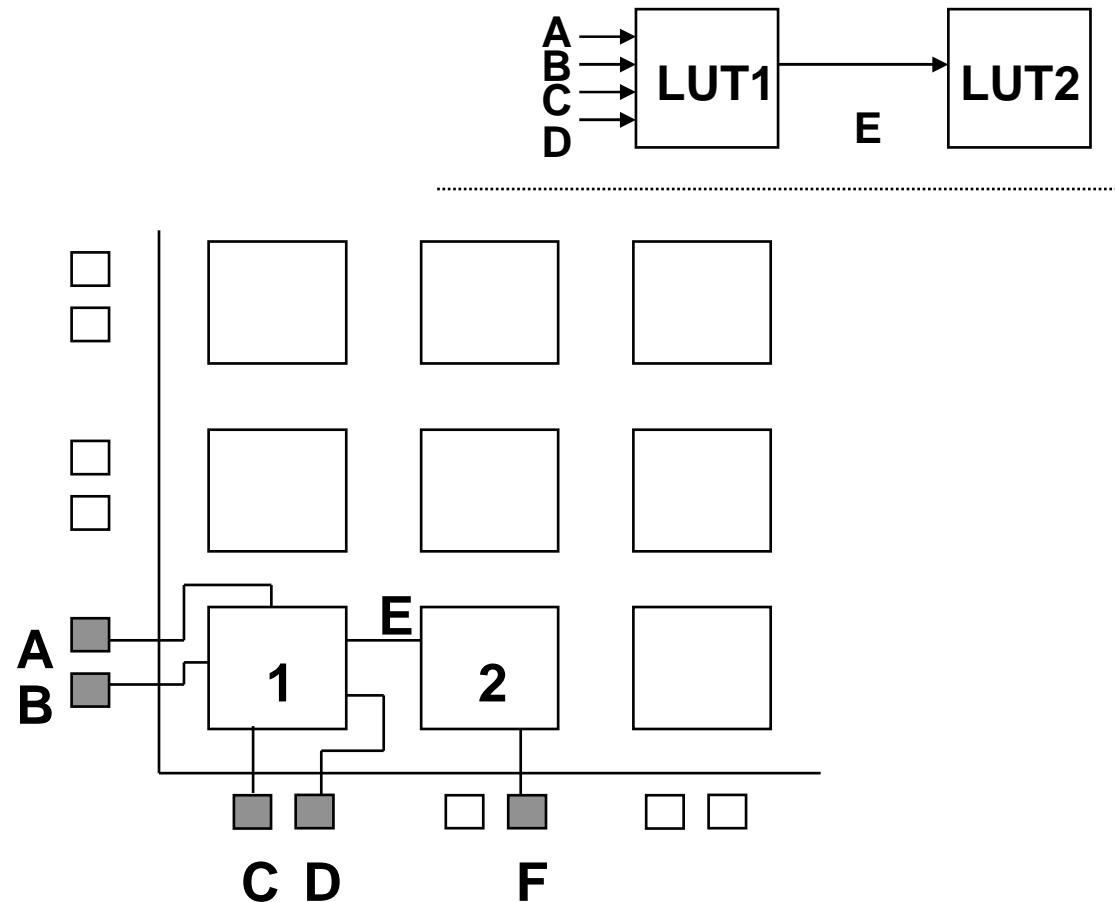
Wire length measures

- Estimate wire length by distance between components.
- Possible distance measures:
 - Euclidean distance ($\sqrt{x^2 + y^2}$);
 - Manhattan distance ($x + y$).
- Multi-point nets must be broken up into trees for good estimates.



Placement

- Placement has a set of competing goals.
- Can't optimize locally and globally simultaneously.
- Use heuristic approaches to evaluate quality.



Placement Algorithms

- **Constructive methods: begin from netlist and generate an initial placement.**
 - **Partitioning methods: mincut and Kernighan-Lin methods**
 - **Clustering**
- **Iterative improvement**
 - **Begin with random or constructive placement.**
 - **Iterate to improve it.**
 - **Hill-climbing**

Iterative Placement Algorithms

- **Pairwise interchange methods**
- **Force-directed methods**
 - **FD relaxation**
 - **FD pairwise exchange**
- **Simulated annealing**
 - **Generates best results**
 - **Can be time consuming**
- **Macro-based approaches**
 - **Genetic algorithms**
 - **Quad swaps**

Iterative Improvement Algorithms

Force-directed: (classical mechanics)

- Force vector computed on each module corresponding to all nets
- Solve set of non-linear differential equations.

Simulated annealing: (statistical mechanics)

- Model a physical annealing process which optimizes energy.
- Similar to “quenching” metal.

Formulating Force Equations

Use Hooke's Law

Modules 1, 2, ... N

m_i mass of module i

x_i x position of module i

K_{ij} Attractive constant between module i and j

F_i Net force on module i from rest of modules

$$\frac{d^2 x_i}{dt^2} = F_i = - \sum_{j=1}^N K_{ij} (x_i - x_j)$$

Force-directed (cont.)

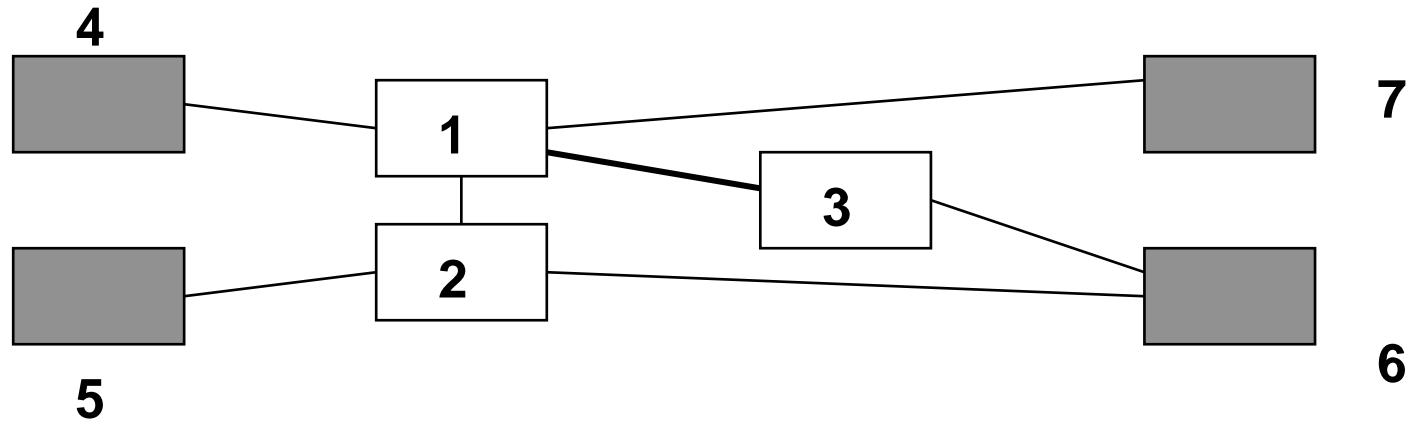
- We know that for the steady state

$$d^2X_i / dT^2 = 0$$

- Determine set of non-linear equations and solve simultaneously using Newton's Method.
- Problem size can grow quite large as size of device increases.

- Interaction between X and Y coords
 - 3D Device?

Example



$$X_4 = X_5 = 0$$

$$X_6 = X_7 = L$$

$$0 = R [1/X_1 + 1/X_1-L + 1/X_1-X_2 + 1/X_1-X_3] \\ - [X_1 + (X_1-L) + 3(X_1-X_3) + (X_1-X_2)]$$

- Different results for different R values
- Solve equations simultaneously

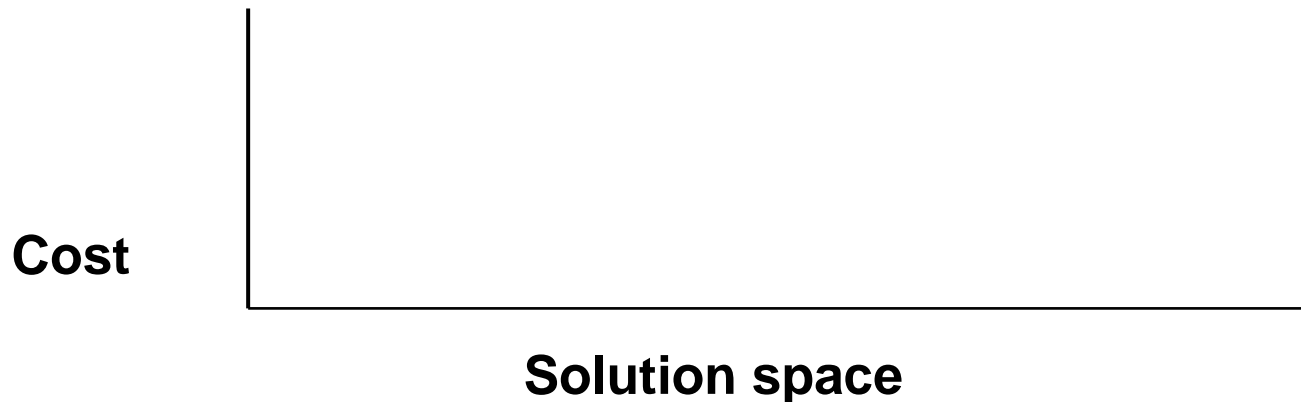
Both for X and Y -> different repulsive forces?

Force-Directed Relaxation

- Start with random placement.
 - Compute forces on each module.
 - Pick the module with the largest force on it
 - Compute zero-force position with Newton's method
 - Attempt to move to unoccupied position
 - Or swap with existing module
 - Or move to nearest open position
- **Continue until final locations determined.**

Hill Climbing Algorithms

- To avoid getting trapped in local minima, consider “hill-climbing” approach
- Need to accept worse solutions or make “bad” moves to get global minima.
- Acceptance is probabilistic. Only accept cost-increasing moves some of the time.



Physical Annealing

- Take a metal and heat to high temperature
- Allow it to cool slowly; metal is annealed to a low temperature
- Atoms in the metal are at lower energy states after annealing
- Higher the temperature initially and slower the cooling, the tougher the metal becomes.
- Atoms transition to high energy states and then move to low energy.

Simulated Annealing

- Optimization strategy based on physical annealing process
- Generate random moves.
 - Initially, accept moves that decrease and increase cost.
- As temperature decreases, the probability of accepting bad moves decreases.
- Eventually, default to greedy algorithm

Only accept positive moves

Determine when to terminate.

Annealing Algorithm

T = StartingT

Moves_per_iteration = $BN^{4/3}$

```
While (stopping_criteria(T) = false) {  
    While (Move_Count < Moves_per_Iter) {  
        swap blocks  
        evaluate  $\Delta$ cost  
        if (Accept <  $\Delta$ cost)  
            Move block to new location  
    }  
    T = update(T)  
}
```

N = # blocks

B = scaling factor

T = temperature

Accept Function

$\Delta\text{cost} = \text{new cost} - \text{initial cost}$

If ($\Delta\text{cost} \leq 0$)

 return (yes)

Else {

$y = \exp(-\Delta\text{cost}/T)$

$r = \text{random}(0, 1)$

 if ($r < y$)

 return (yes)

 else

 return (no)

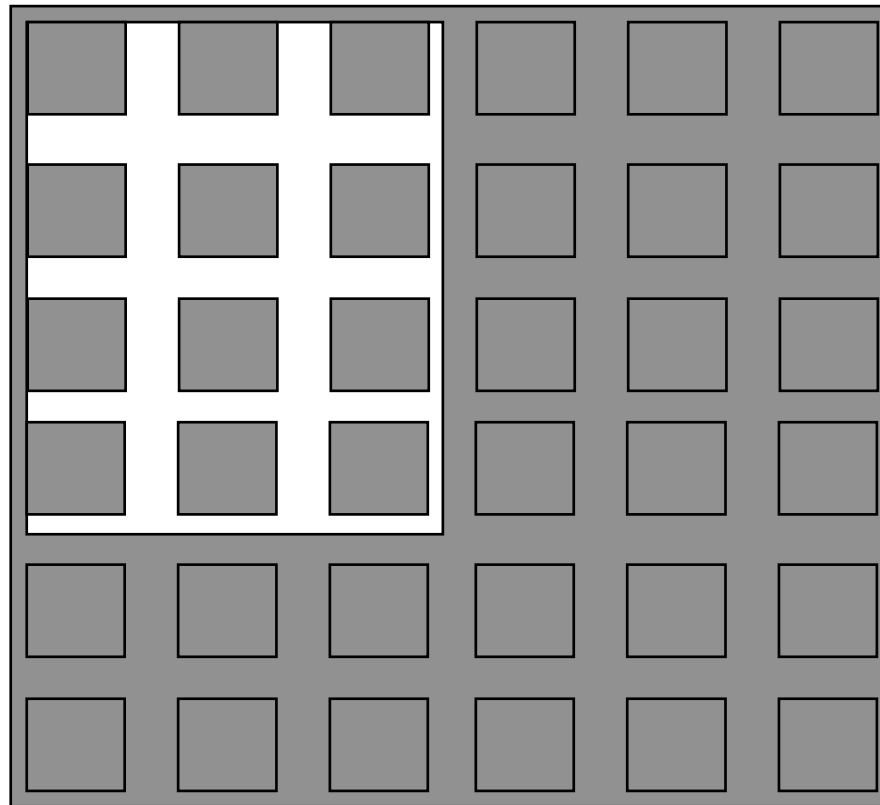
}

Annealing Criteria

- Contemporary FPGA packages use the following parameters:
 1. Starting temp – $20 * \text{stand_dev}(\text{cost of } N \text{ swaps})$
 2. Cost function – weighted sum of wire length and delay
 3. Inner loop – $B * N^{4/3}$
 - Beta cost function
 4. Stopping criteria –
 - $T < [.005 * \text{Cost}/N_{\text{nets}}]$

Range Limiting

- As temperature drops, limit scope of swaps
- Increased likelihood of acceptance.
- Can also used to secure critical path performance.



Timing-driven Placement

- Take both wire length and critical path into account
- Problem
 - Critical path changes as I move blocks
 - How do I balance the two objectives
- How do we go about modeling routing delay during placement?

$$Timing_Cost(i,j) = Delay(i,j) \cdot Criticality(i,j)^{Criticality_Exponent}$$

Estimating delays

- Marquardt and Betz approach (T-VPlace)
 - Perform initial delay analysis to determine shortest delays between each pair of X, Y locations
 - Store information in table for quick look-up
 - Assumption that router will probably find the minimum delay path (a leap of faith!)

$$Timing_Cost(i,j) = Delay(i,j) \cdot Criticality(i,j)^{Criticality_Exponent}$$

Determining Criticality

- Same basic approach as used for clustering criticality
- For each (i, j) connection from source i and sink j
 - Determine arrival times (pre-order BFS)
 - Determine *required* arrival times (post-order BFS)
 - Determine slack -> required_arrival_time – arrival_time
 - Criticality(i, j) = [1- slack(i, j)]/ (Max slack)

$$Timing_Cost(i,j) = Delay(i,j) \cdot Criticality(i,j)^{Criticality_Exponent}$$

What is the purpose of the criticality exponent?

Balancing Wiring and Timing Cost

- Need to determine relative changes in timing and wiring based on moves

$$\Delta C = \lambda \cdot \frac{\Delta \text{Timing_Cost}}{\text{Previous_Timing_Cost}} + (1 - \lambda) \cdot \frac{\Delta \text{Wiring_Cost}}{\text{Previous_Wiring_Cost}}$$

- Idea: Use relative changes from previous calculation
 - Both values less than 1
 - Helps balance effect based on scaling parameter

This still doesn't help address changes in delay

Updated Annealing Algorithm

```
S = RandomPlacement ();
T = InitialTemperature ();
Rlimit = InitialRlimit ();
Criticality_Exponent = ComputeNewExponent();

ComputeDelayMatrix();

while (ExitCriterion () == False) {    /* "Outer loop" */

    TimingAnalyze();    /* Perform a timing-analysis and update each connections criticality */
    Previous_Wiring_Cost = Wiring_Cost(S);    /* wire-length minimization normalization term */
    Previous_Timing_Cost = Timing_Cost(S);    /* delay minimization normalization term */

    while (InnerLoopCriterion () == False) {    /* "Inner loop" */

        Snew = GenerateViaMove (S, Rlimit);
        ΔTiming_Cost = Timing_Cost(Snew) - Timing_Cost(S);
        ΔWiring_Cost = Wiring_Cost(Snew) - Wiring_Cost(S);
        ΔC = λ·(ΔTiming_Cost/Prev_Timing_Cost) +
              (1-λ)·(ΔWiring_Cost/Previous_Wiring_Cost); /* new cost fcn */

        if (ΔC < 0) {
            S = Snew /* Move is good, accept */
        }
        else {
            r = random (0,1);
            if (r < e-ΔC/T) {
                S = Snew; /* Move is bad, accept anyway */
            }
        }
    }    /* End "inner loop" */

    T = UpdateTemp ();
    Rlimit = UpdateRlimit ();
    Criticality_Exponent = ComputeNewExponent();

}    /* End "outer loop" */
```

How often to recalculate delay?

- Recalculating delay once per temperature is good.
- Also simplifies programming somewhat

TABLE 1.1 Effect of timing-analysis in the outer loop

Timing-Analysis Interval	Placement Estimated Critical Path (ns) (20 Circuit Geometric Average)	Wiring Cost (20 Circuit Geometric Average)
1	39.3	529.6
2	39.5	531.1
4	40.1	530.5
8	40.5	531.0
16	39.5	530.3
32	41.4	534.5
64	41.3	528.3
128	43.0	522.9
Never	43.0	522.9

How important is timing-driven placement?

TABLE 1.6 Post-place-and-route comparison of VPlace and T-VPlace (cluster size = 1).

Circuit	Post-Place-and-Route Minimum Channel Width (W_{\min})			Post-Place-and-Route Critical Path (ns) $W = \infty$			Post-Place-and-Route Critical Path (ns) $W = W_{\min} + 20\%$		
	VPlace	T-VPlace ($\lambda = 0$)	T-VPlace ($\lambda = 0.5$)	VPlace	T-VPlace ($\lambda = 0$)	T-VPlace ($\lambda = 0.5$)	VPlace	T-VPlace ($\lambda = 0$)	T-VPlace ($\lambda = 0.5$)
alu4	14	14	14	40.3	40.4	29.8	42.4	41.2	33.4
apex2	15	17	16	46.9	46.3	32.3	47.7	46.5	48.8
apex4	17	16	18	40.9	44.8	28.2	42.0	46.8	31.7
bigkey	13	13	10	36.0	35.2	21.6	36.7	35.4	25.2
clma	16	16	17	90.2	91.1	72.3	116.0	166.0	130.0
des	11	12	11	40.5	48.9	30.2	50.4	57.4	43.7
diffeq	11	11	12	35.2	37.5	30.8	38.9	41.0	34.9
dsip	12	12	12	27.9	27.2	21.7	28.3	28.8	22.9
elliptic	14	16	15	70.6	76.1	46.1	79.5	79.6	58.1
ex1010	14	15	15	85.0	77.5	52.9	96.2	78.6	70.5
ex5p	17	17	19	39.6	40.4	28.1	42.7	42.7	43.5
frisc	16	17	18	70.8	73.2	59.6	76.8	79.6	61.6
misex3	14	15	15	39.0	40.2	26.6	39.3	75.0	34.3
pdc	22	21	24	81.7	74.5	49.9	122.0	114.0	73.0
s298	11	12	12	74.8	72.0	53.6	116.0	78.7	77.8
s38417	11	11	12	61.7	71.0	33.7	70.0	74.6	37.2
s38584.1	11	11	11	45.3	44.1	31.8	49.7	44.3	36.4
seq	16	16	16	45.7	41.0	28.1	46.4	43.7	39.5
spla	18	18	20	58.4	67.4	39.7	74.8	100.0	69.4
tseng	9	10	11	33.7	33.1	28.3	39.8	38.4	33.1
Geom. Av.	13.78	14.22	14.50	50.1	51.0	35.2	57.1	59.2	45.7
%diff w.r.t VPlace	—	+3.2%	+5.2%	—	+1.8%	-29.7%	—	+1.04%	-20.0%

Run time
Penalty –

2.5X

Summary

- **Placement and clustering of modules critically important for subsequent routing step**
- **Often initial placement performed and then iteratively improved**
- **Mincut partitioning approaches sometimes used for initial placement**
- **Efficient timing analysis a key to successful placement**
- **Island-style devices benefit from simulated annealing approaches**
 - **Accurate cost function is the key to success**
- **Issues related to power consumption remain**