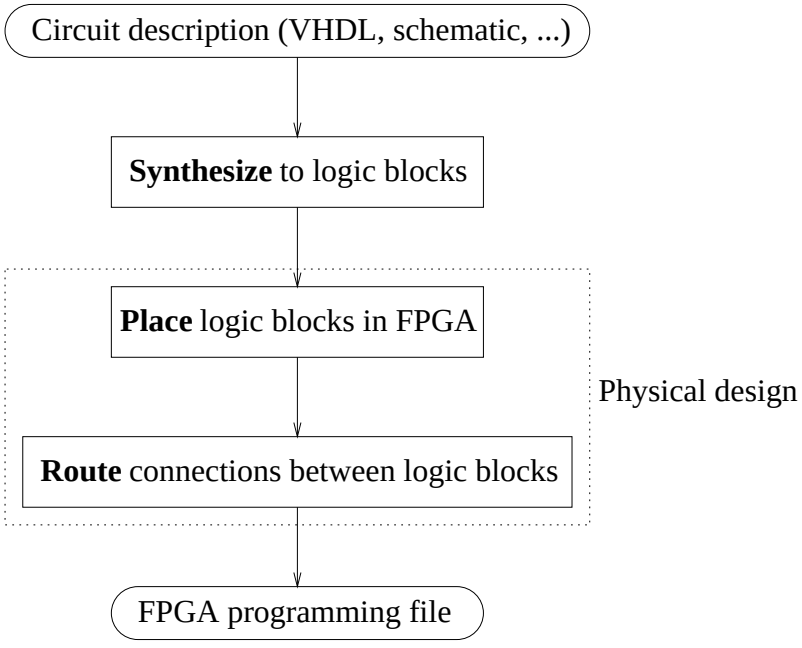
Place and Route for FPGAs

FPGA CAD Flow



Placement

Goal: Determine which logic block within an FPGA should implement each of the logic blocks required by the circuit.

Objective: Minimize the required wiring (wire-length driven placement).
Balance the wiring density across the FPGA (routability-driven placement).
Maximize circuit speed (timing-driven placement).

- 3 major classes of placers:
  - min-cut (partitioning-based) placers
  - analytic placers
  - simulated annealing based placers.

Simulated Annealing

- Simulated annealing (SA) mimics the annealing process used to gradually cool molten metal to produce a high-quality crystal structure.
- SA takes an existing solution and then makes successive changes in a series of random moves.
- Each move is accepted or rejected based on an *energy function*.
- In order to escape from solutions which are *local minima*, it allows "up-hill" moves that seemingly move to a less desirable solution.
- Probability of accepting an "up-hill" move is controlled by the expression $e^{\frac{-\triangle E}{T}}$ where $\triangle E$ is the resulting increase in the energy function and T is the current *temperature*.
- The temperature is slowly decreased and the system finally comes to rest at a low-energy configuration.

Simulated Annealing Based Placer

- An initial placement is created by assigning logic blocks of the circuit randomly to the available locations in the FPGA.
- A common cost function in wirelength-driven placement is the sum over all nets of the half-perimeter of their bounding boxes.
- A move can be the exchange of locations of two randomly selected logic blocks.
- Initially, temperature T is very high so almost all moves are accepted; it is gradually decreased as the placement is refined so that eventually the probability of accepting a move that makes the placement worst is very low.
- An advantage of simulated annealing based placer is the ease to add new optimization objectives or constraints. But a drawback is the running time required.

Algorithm: Simulated Annealing Based Placer

1 **begin**
2 $S =$ RandomPlacement();
3 $T =$ InitialTemperature();
4 **while** (ExitCriterion() == False) **do**
5 **while** (InnerLoopCriterion() == False) **do**
6 $S' =$ GenerateViaMove(S);
7 $\Delta \leftarrow cost(S') - cost(S)$;
8 **if** $\Delta \leq 0$ **then** $S \leftarrow S'$
9 **if** $\Delta > 0$ **then** $S \leftarrow S'$ with probability $e^{-\frac{\Delta}{T}}$;
10 **end while**;
11 $T =$ UpdateTemp();
12 **end while**;
13 **return** S
14 **end**

Min-Cut Placement

- The given circuit is repeatedly partitioned into two subcircuits.
- Meanwhile, the chip area is partitioned alternately in the horizontal and vertical directions into subsections.
- Each subcircuit is assigned to a subsection.
- The process is repeated until each subcircuit consists of a single logic block and has a unique location on the chip area.
- During partitioning, the number of nets that are cut by the partition is usually minimized based on the intuition that densely connected subcircuits should be placed closely.

Partition Methods

- A commonly used partition approach is the iterative improvement approach that makes local changes to an initial partition.
- Kernighan and Lin (K-L) proposed a graph bisectioning algorithm which starts with a random initial partition and then uses pairwise swapping of vertices between partitions, until no improvement is possible.
- A simple *greedy algorithm* will accept a move only if it provides immediate benefit, but such shortsightedness often leads to a local minimum from which the algorithm cannot escape.
- The K-L algorithm was designed to be able to escape from local minima.

Algorithm: Kernighan-Lin(G)
Input: $G = (V, E), |V| = 2n$.
Output: A bisectioning of G into A and B with "small" cut cost.

1 **begin**
2 Bipartition G into A and B such that $|V_A| = |V_B|$, $V_A \cap V_B = \emptyset$, and $V_A \cup V_B = V$.
3 **repeat**
4 Compute D_v, $\forall v \in V$. /* D_v = cost reduction for moving v */
5 **for** $i = 1$ **to** n **do**
6 Find a pair of unlocked vertices $v_{ai} \in V_A$ and $v_{bi} \in V_B$ whose exchange makes the largest decrease or smallest increase in cut cost;
7 Mark v_{ai} and v_{bi} as locked, store the gain $\hat{g}_i$, and compute the new D_v, for all unlocked $v \in V$;
8 **end for**;
9 Find k, such that $G_k = \sum_{i=1}^{k} \hat{g}_i$ is maximized;
10 **if** $G_k > 0$ **then**
11 Move $v_{a1}, \ldots, v_{ak}$ from V_A to V_B and $v_{b1}, \ldots, v_{bk}$ from V_B to V_A;
12 Unlock v, $\forall v \in V$.
13 **until** $G_k \leq 0$;
14 **end**

Kernighan-Lin Algorithm

- One pass of the K-L algorithm:
  - Vertex pairs which give the largest decrease or the smallest increase in cut size are exchanged.
  - These vertices are then *locked* (thus are prohibited from participating in any further exchange).
  - The process is continued until all the vertices are locked.
- If there is no k such that $G_k > 0$ then the current partition (A, B) cannot be improved by pairwise swapping and the algorithm terminates. Otherwise, we choose the k that maximizes G_k and make the interchange of $\{v_{a1}, \ldots, v_{ak}\}$ and $\{v_{b1}, \ldots, v_{bk}\}$ permanent and perform another pass of the K-L algorithm.

Fiduccia-Mattheyses Algorithm

- The K-L algorithm partitions a circuit modelled as a graph such that the cost of the edges cut the partition is minimized. Fiduccia and Mattheyses (F-M) takes into consideration multiple-terminal nets and the sizes of the circuit elements.

- Features of F-M that are different from K-L
  - Reduce the number of *nets* cut instead of the number of *edge* cut.
  - Only a single vertex is moved across the cut in a single move.
  - Vertices can have different sizes.
  - Can handle uneven sized partitions; a balance factor is introduced.
  - A special data structure is used to select vertices to be moved across the cut to improve running time.

Analytic Placers

- Analytic placers make use of numerical technique to compute the "coordinates" of the logic blocks.

- It is often based on a *quadratic wirelength* objective.

- If a *connectivity matrix* $C = [c_{ij}]$ represents the connection between the logic blocks of the circuit, then minimizing the quadratic wirelength is to minimize

$$\frac{1}{2}\sum_{i,j=1}^{n} c_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2]$$

  where (x_i, y_i) is the coordinates of logic block i on the layout area.

- The quadratic wirelength objective can be rewritten using matrix notation as

$$\vec{x}^T B\vec{x} + \vec{y}^T B\vec{y}$$

  where $B = [b_{ij}]$ has entry b_{ij} equal to $-c_{ij}$ for $i \neq j$ and diagonal entry b_{ii} equal to $\sum_{j=1}^{n} c_{ij}$.

- Numerical technique is well-suited for optimization of the above form of objective.

FPGA-Specific Placement Issues

- The number of routing tracks in routing channels are fixed on a FPGA.
- A necessary condition for any feasible placement solution is the *channel density* in every channel cannot exceed the number of routing tracks available in the channel.
- In order to calculate the channel density accurately, some SA-based placement algorithms also perform *global routing* iteratively for every move.
- A FPGA contains routing tracks of various lengths.
- Simple interconnection delay estimation model based on net length or fanout are not accurate enough for use in timing-driven FPGA placement algorithms.
- Fast and accurate interconnection delay computation methods are needed for timing-driven FPGA placement.

Routing

- Once the locations for all the logic blocks in a circuit have been chosen, a router assigns the nets of the circuit to the routing segments on the FPGA and determine which programmable switches should be turned on to connect the logic blocks as required by the circuit.
- Because of the high complexity involved in routing, routing is usually performed in two phases: *global routing* and *detailed routing*.

Global Routing

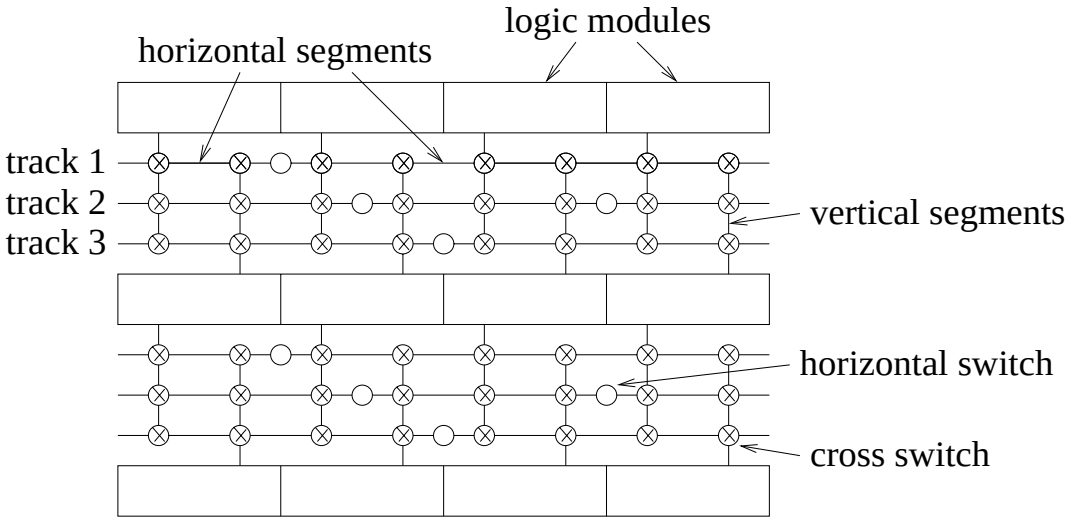
- Global routing finds for each net a routing tree by selecting a set of routing channels, but does not choose specific routing tracks and switches for each net.
- Since routability is the most important issue, minimization of *channel density* is normally the optimization objective in FPGA global routing.
- To estimate the routability more accurately, the connectivity architectural details within the channel intersection areas also need to be considered because channel intersection areas are usually not fully populated with switches.

Detailed Routing

- Detailed routing assigns each net to specific routing segments in the channels as restricted by the global router.
- Design of detailed routing algorithms heavily depends on the FPGA routing architecture. For example, detailed routing algorithms for row-based and symmetrical array based architectures are significantly different.

Detailed Routing for Row-Based Architecture

- Routing channels in row-based architectures are segmented.
- A routing track in the segmented channel is divided into several segments of various lengths with switches between the adjacent routing segments.
- In the row-based architecture, crossings between vertical and horizontal routing segments are fully populated with switches so that any vertical routing segments can be connected to any crossing horizontal routing segment where necessary.
- So detailed routing in row-based architecture is reduced to the *segmented channel routing problem*.



A row-based routing architecture. The crossings between vertical and horizontal routing segments are fully populated with switches.

- Because switches can introduce significant delay due to the relatively high fuse resistance, the number of switches allowed for completing a net connection is usually restricted in order to achieve high circuit performance.

- *K-segment channel routing* requires that the maximum number of segments used for routing any net connection be limited to K.

- For $K = 1$, the K-segment channel routing problem can be solved efficiently using bipartite matching methods. However, for $K \geq 1$, the problem becomes NP-hard.

- Efficient heuristic algorithms are used to solve the general segmented channel routing problem.

Detailed Routing for Symmetrical-Array Architecture

- The crossing vertical and horizontal routing tracks in symmetrical array based architecture usually are not fully populated with switches.

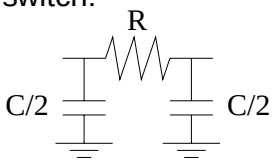
- Hence detailed routing for symmetrical array based architecture cannot be reduced to individual channel routing problems.

- A commonly used approach is to explore the connectivity within the routing channels specified by the global router by using some path searching technique.

- Contention for limited routing resources between different nets almost always necessitates rip-up and reroute in the process.

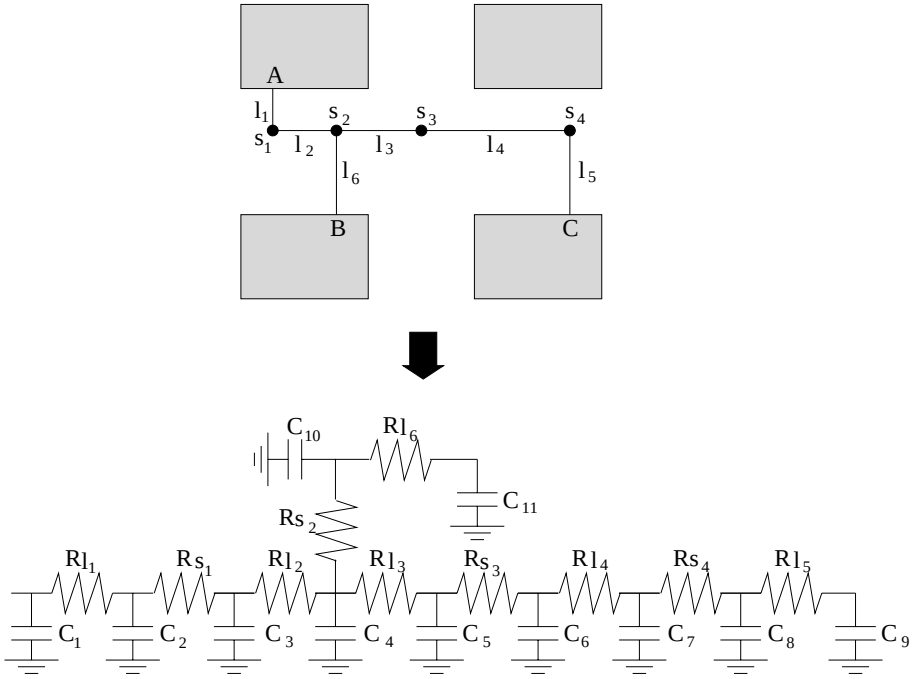
Interconnect Delay Estimation

- The capacitance and resistance of switches/wires are the main cause of interconnect delay.
- Researchers have modelled a net's routing as a RC-tree.
- The π-model for a wire or an "on" switch:



- The Elmore delay of a source-sink path P is

$$\sum_{i \text{ on } P} R_i \cdot C(subtree_i)$$



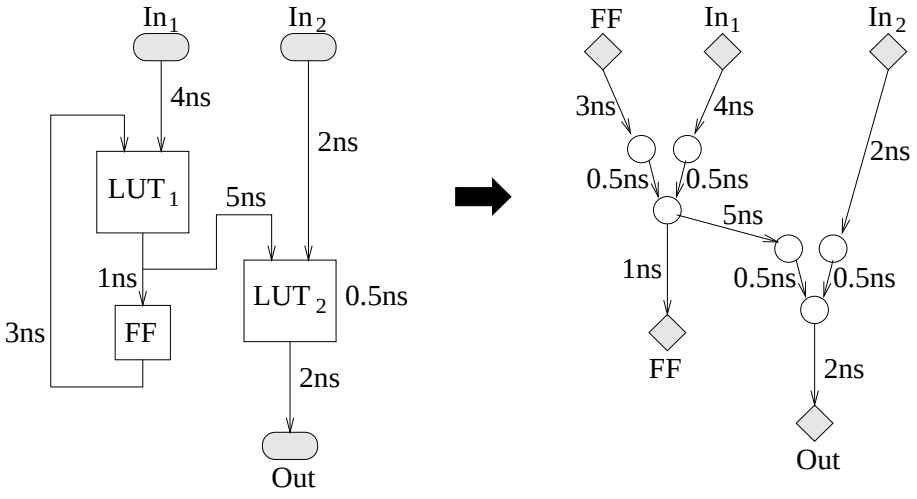
A RC-tree for a net's routing.

Timing Analysis

- Timing analysis can determine how fast a circuit will operate and help to guide timing-driven CAD tools.
- The longest path delay among all combinational paths dictates how fast a circuit can operate.
- A *timing graph* can be used in timing analysis.
- The minimum required clock period can be computed in $O(n)$ time for a timing graph with n nodes by traversing the nodes in topological order.
- First, nodes with no incoming edges are labelled with an arrival time of 0. A node i can be labelled if all nodes j from which it receives input have been labelled.

$$T_{arrival}(i) = max_{j \in fanin(i)}\{T_{arrival}(j) + delay(j,i)\}$$

- The largest arrival time D_{max} of all nodes is the minimum clock period required.



A timing graph for a simple circuit.

- The *slack* of a connection is the amount of delay that could be added to this connection without increasing the minimum cycle time of the circuit.

- The slack of each connection can be computed to guide timing-driven placement and routing tools.

- The required times of the nodes can be computed by traversing the nodes in reverse topological order.

- First, nodes with no outgoing edges are labelled with a required time of D_{max}. The required time of any node with fanout is computed according to

$$T_{required}(i) = min_{j \in fanout(i)}\{T_{required}(j) - delay(i,j)\}$$

- The slack of a connection from node i to node j is

$$slack(i,j) = T_{required}(j) - T_{arrival}(i) - delay(i,j)$$

- Connections with a zero slack are on the circuit critical path(s).

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.