Floorplan design of VLSI circuits using Simulated Annealing

Gaurav Rajasekar

Muneeb Zia

Introduction

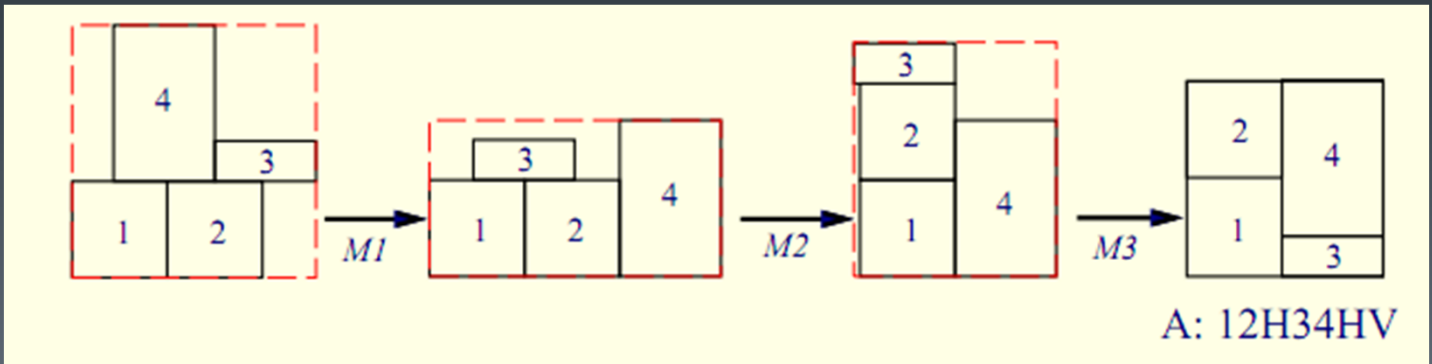
- Initial floor-plan is represented by a string called the Polish Expression.
- Eg.
  - E = 16H2V75VH34HV
- A Normalized Polish Expression is one in which there are no consecutive operators of the same type (H or V respectively).
- Enables construction of a unique slicing floor-plan.

Problem Formulation

- Normalized Polish Expression string given.
- A ‘cost’ function needs to be calculated and minimized.
  - Ø = A + λW.
  - A: area of the smallest rectangle
  - W: overall wiring length
  - λ: user-specified parameter (in our case it’s 0)
- An iterative process needs to be implemented to introduce perturbations into the Polish Expression and ‘anneal’ it in a process analogous to how a metal is annealed.
- Language used : C++

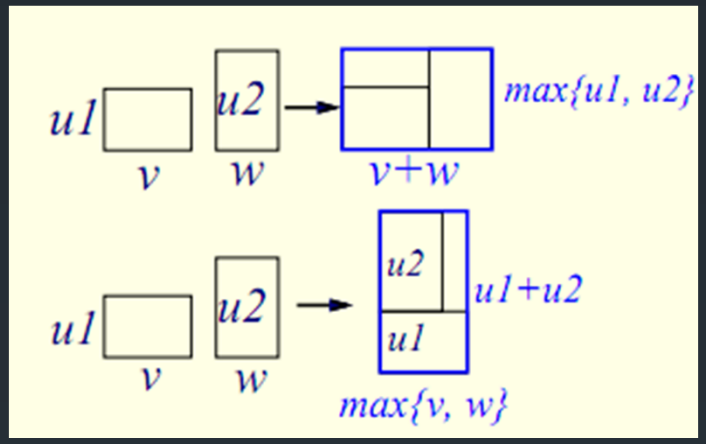
Types of Moves

- Three types of moves:
  - M1 (Operand Swap): Swap two adjacent operands in the polish expression.
  - M2 (Chain Invert): Complement a chain in the polish expression.
  - M3 (Operator/Operand Swap): Swap two adjacent operands and operators.
- Balloting property is maintained during M1 and M2 moves but it may be violated during the M3 move.

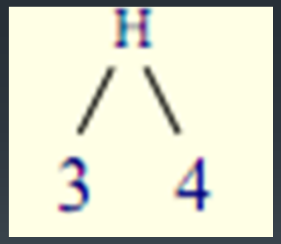


Placement on Floorplan

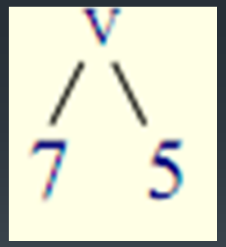
- Width and Height calculation:



- Block Placement on the Floorplan:



- 3 on top and 4 below it.



7 on the left and 5 on the right.

Implementation

- Language used : C++

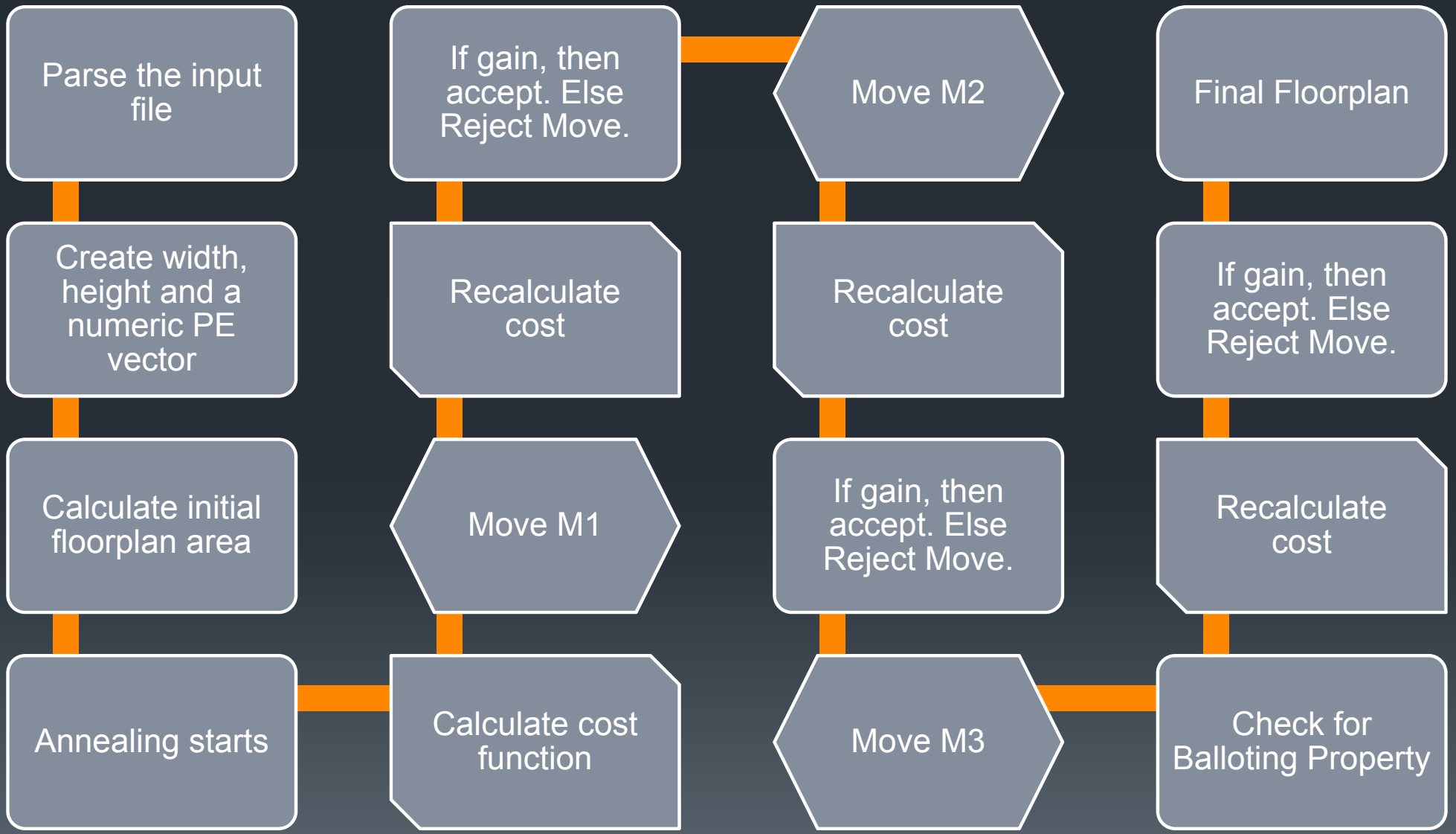
- Data Structure used: Vector
  - We use a vector each to store the input Polish Expression, the widths of the blocks and the heights of the blocks.

  - Why?
    - Vector functions are easier in C++ than operating on structs and tree nodes.
    - Easier to add, swap or delete nodes.
    - Struct usage would have required a boolean operator to tell you if it's a numerical / H / V node.
    - Less memory used for each node.
    - Here the P.E. is stored in a vector of integers not as characters.

Input Parsing

- Preprocessing involves separating the nodes from the hyphenated string and passing it into a vector.
- The H and V nodes are replaced by two arbitrarily chosen negative numbers -4 and -7 to differentiate them from the rest of the numbered nodes and to keep the vector elements to an integer data-type only.
- The widths and heights are segregated and put into 2 separate vectors.
- Widths and heights are arranged in increasing numeric order i.e. width of 0^{th} block is first and so on.

Process


Parse the input file
Create width, height and a numeric PE vector
Calculate initial floorplan area
Annealing starts
Calculate cost function
Move M1
Recalculate cost
If gain, then accept. Else Reject Move.
Move M2
Recalculate cost
If gain, then accept. Else Reject Move.
Move M3
Check for Balloting Property
Recalculate cost
If gain, then accept. Else Reject Move.
Final Floorplan

Annealing Process

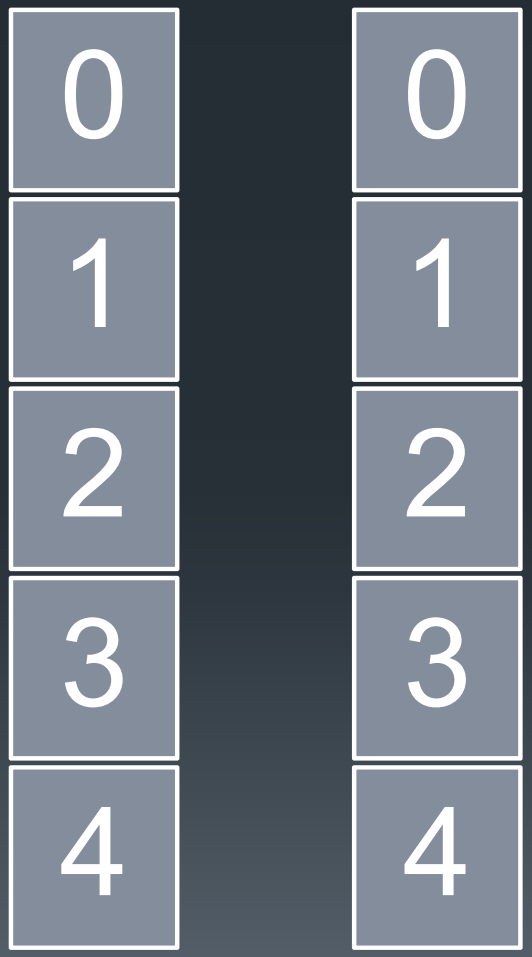
- While rejection probability < 95%
  AND
  Temperature is greater than the threshold provided
  - While uphill < N where N = k*n
    AND
    Moves tried (MT) < 2*N
    - Random moves between M1, M2 and M3 are chosen and Polish Expression is modified.
    - Cost for each new expression is calculated.
      - If ΔCost < 0 then New Expression = Best
      - Else Reject the move and increment an uphill counter
  - Cool. ;)

Area computation

- 2-1-0-H-V-3-V-4-V

- Look for the first H or V, then perform the respective operation for H or V for the previous 2 entries in the Polish Expression.
- Remove the 3 nodes and replace them by a single new block.
- Append new block's width and height at the end of the width and height vectors.

Width & Height Vectors



Area computation

- 2-1-0-H-V-3-V-4-V

- 2-5-V-3-V-4-V

Width & Height Vectors

0	0
1	1
2	2
3	3
4	4
5	5

Area computation

- 2-5-V-3-V-4-V

- 6-3-V-4-V

Width & Height Vectors

0	0
1	1
2	2
3	3
4	4
5	5
6	6

Area computation

- 6-3-V-4-V

- 7-4-V

Width & Height Vectors

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7

Area computation

- 7-4-V

- 8

- Wherein 8 is the overall floorplan area.

Width & Height Vectors

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	**8**

Results

Circuit	Initial Area	Minimum Final Area after Annealing
5_block.ple	65	55
10_block.ple	147	119
30_block.ple	1075	903
100_block.ple	7119	6592
150_block.ple	14104	13114

Conclusion

- The runtime depends on the parameters that we're supplying:
  - Cooling Ratio
  - Epsilon (Minimum Temperature)
  - 'k' limits the moves tried
  - Number of Iterations (Limits runtime)

- Our code seems to give greater floorplan area gains in larger designs.

Possible Extensions

- We could include a constraint on our floorplan's aspect ratio.
- If we could get a HotSpot model of the floorplan we could include temperature of the blocks as one of the criteria during calculation of the cost function to perform temperature aware floorplanning of circuits.
  - Add Maximum temperature to the objective function.
- Incorporating Genetic Algorithm
  - The Crossover operations of GA can enable faster searching of a wider solution space than what's possible by using Simulated Annealing alone.

Y U NO ASK QUESTIONS?