

**INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**

**NPTEL**

**NPTEL ONLINE CERTIFICATION COURSE  
An Initiative of MHRD**

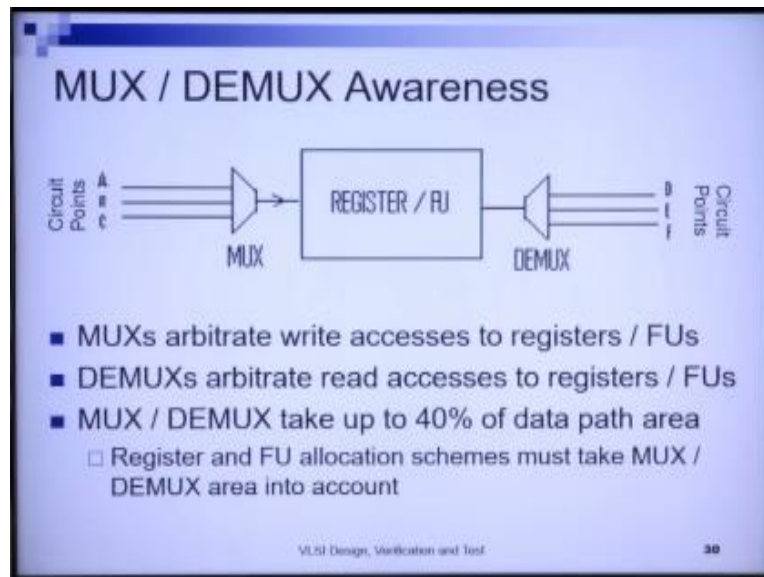
**VLSI Design, Verification & Test**

**Dr. Arnab Sarkar  
Department of CSE  
IIT Guwahati**

Hello welcome to module three of lecture seven in the last module we look at resource sharing and binding problems with respect to single type of resources we looked at the register allocation and minimization problem we looked at the functional unit allocation and minimization problem we saw that for simple data flow graph within an operation constraints craft such allocation can be and we optimally done in polynomial time by their corresponding interval conflict cross and using the left edge algorithm.

And we also saw that for certain cases when we have loops and branches across operation constraints graph the allocation the resource allocation problem cannot be back to simple interval graphs and hence their conflict graph coloring problems also become so NP complete and therefore enumerative techniques are required to learn to solve them but we used heuristic technique are furiously graph coloring technique to solve that problem however in the last lecture we dealt with the allocation and mapping possible resource styles.

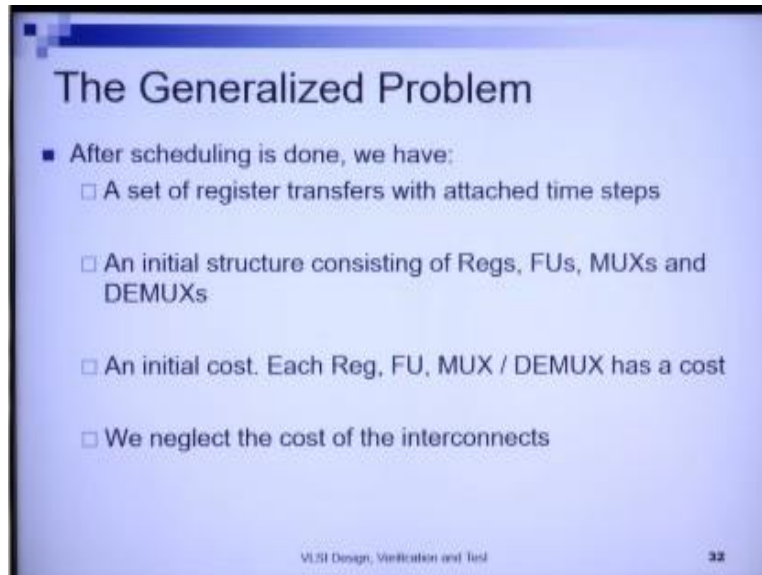
(Refer Slide Time: 01:50)



In this lecture we will deal with the generalized resource allocation problems in the introduction to resource allocation and binding we said that a certain choice of resource allocation effects further dependent resource allocations for example we said that the register and function allocation choice will also determine the MUX choice why because the numbers of and types of functional units that we have as well as registers that we have will determine how many MUX / DEMUX and what will be their type of MUX and DEMUX will be determined by that choice.

And we said that MUX arbitrate write accesses to registers or functional units for example here we have three circuit components circuit points which flowed their output on the registers land MUXs and the outputs of these registers and MUXs again go to circuit components DDF right and the DEMUX arbitrate read accesses to registers or functional units and MUX DEMAX take up to 40% of the data path and hence their allocation problems cannot be ignored.

(Refer Slide Time: 03:13)



## The Generalized Problem

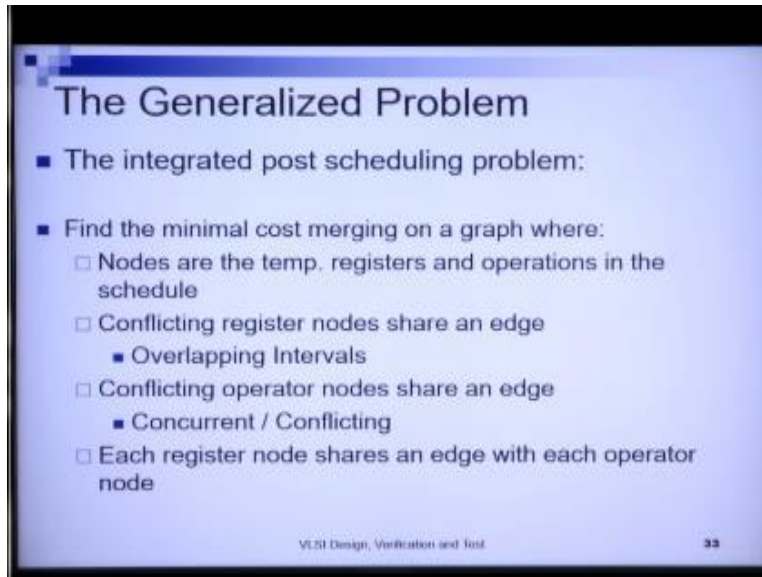
- After scheduling is done, we have:
  - A set of register transfers with attached time steps
  - An initial structure consisting of Regs, FUs, MUXs and DEMUXs
  - An initial cost. Each Reg, FU, MUX / DEMUX has a cost
  - We neglect the cost of the interconnects

VLSI Design, Verification and Test 32

With this we come to the generalized problem the generalized post shading problem may be posed as follows after scheduling is done we have a set of register transfers with attached time steps we know an initial structure of the registers functional unit MUXs and DEMUXs say that means we have done an initial register allocation function allocation MUX and DMUX allocation by and say our heuristic graph coloring algorithm that we studied in the last module.

And each register for each register functional unit MUX and DMUX we know it is cost and we currently neglect the cost of interconnects.

(Refer Slide Time: 03:59)



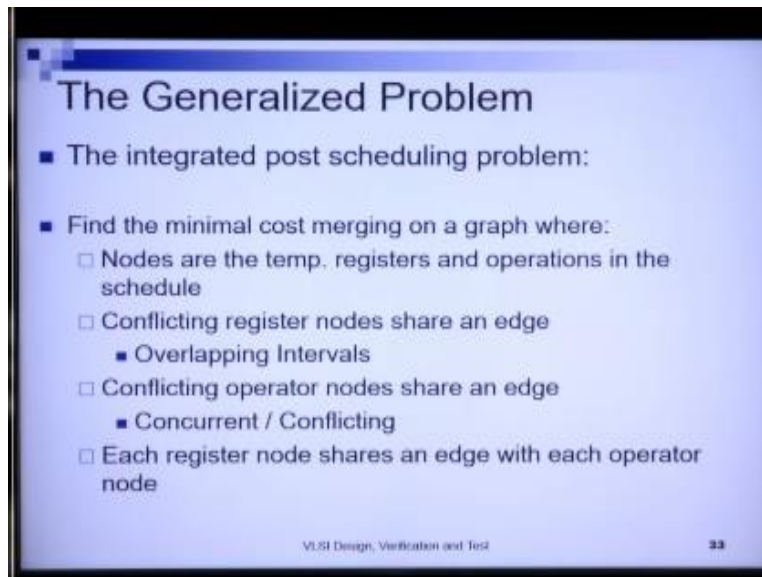
The slide is titled "The Generalized Problem" and contains the following text:

- The integrated post scheduling problem:
  - Find the minimal cost merging on a graph where:
    - Nodes are the temp. registers and operations in the schedule
    - Conflicting register nodes share an edge
      - Overlapping Intervals
    - Conflicting operator nodes share an edge
      - Concurrent / Conflicting
    - Each register node shares an edge with each operator node

At the bottom of the slide, there is a footer that reads "VLSI Design, Verification and Test" on the left and the number "33" on the right.

So given this scenario the integrated post scheduling problem is to find a minimal cost merging on the conflict graph where nodes are the temporal registers and behavioral operations in the schedule conflicting register node share an edge conflicting operation notes this is not operated but operation conflicting operation nodes also share an edge and each register note shares an edge with each range with each operation node.

(Refer Slide Time: 04:27)



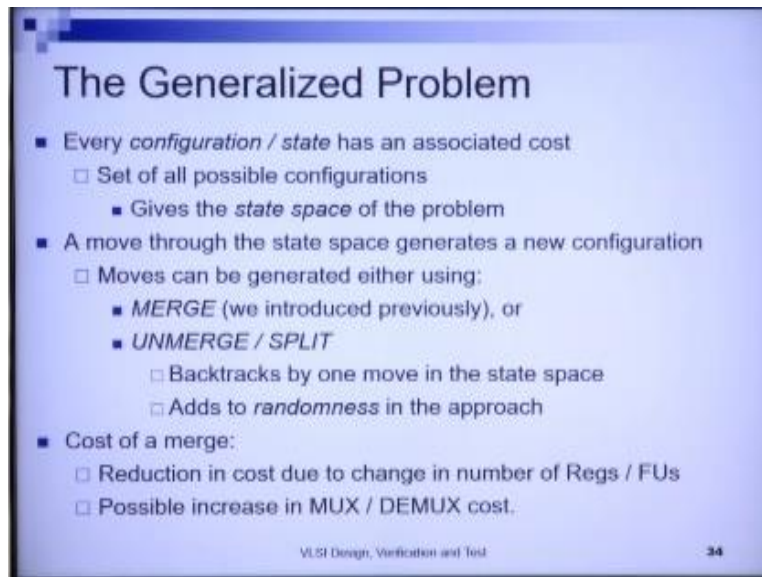
## The Generalized Problem

- The integrated post scheduling problem:
  - Find the minimal cost merging on a graph where:
    - Nodes are the temp. registers and operations in the schedule
    - Conflicting register nodes share an edge
      - Overlapping Intervals
    - Conflicting operator nodes share an edge
      - Concurrent / Conflicting
    - Each register node shares an edge with each operator node

VLSI Design, Verification and Test 33

Right now every configuration or state has an associated cost what is the cost of a configuration a configuration or state in this conflict graph is a given merging of the functional units registers MUX DEMUXs so for a given merging that we have for a given merging that we have we will have a total number of functional units required for allocation or total number of registers required for intervention or total number of MUXs and DEMUXs required for allocation.

(Refer Slide Time: 05:07)



The Generalized Problem

- Every *configuration / state* has an associated cost
  - Set of all possible configurations
    - Gives the *state space* of the problem
- A move through the state space generates a new configuration
  - Moves can be generated either using:
    - *MERGE* (we introduced previously), or
    - *UNMERGE / SPLIT*
      - Backtracks by one move in the state space
      - Adds to *randomness* in the approach
- Cost of a merge:
  - Reduction in cost due to change in number of Regs / FUs
  - Possible increase in MUX / DEMUX cost.

VLSI Design, Verification and Test 34

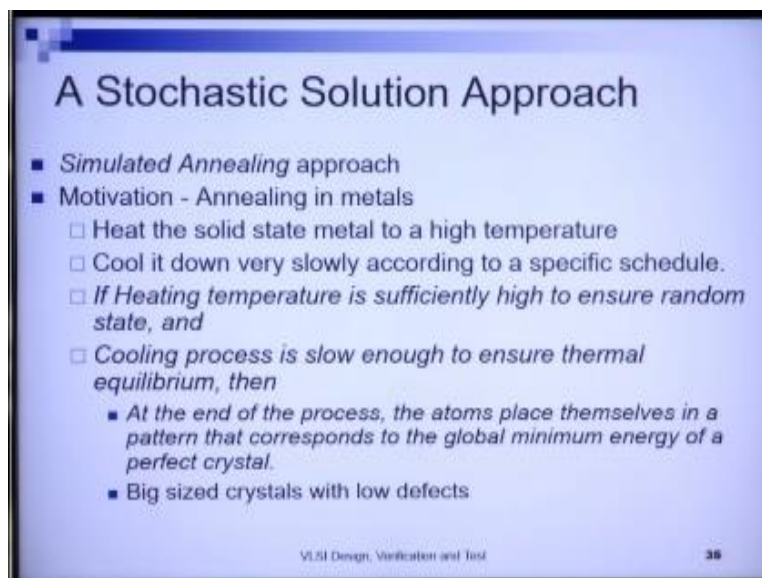
And that gives the cost of the configuration then set of all possible configuration gives the state space of the problem now what the different possible types of merchants that we can do on this conflict graph gives me the gives me the different possible choices that I have and gives a total decent state space of the problem a move through the state space generates a new configuration a move through the same space generates a new configuration and removes can be generated using either emerge which is previously introduced so when we have a merge what do we do we merge two registers together that means that I have allocating the same register instance to two temporary registers.

I am merging two operations together meaning I am allocating the same functional unit instance to these two operations so we merge we introduced previously now we introduce a split a split back tracks by one move in the state space and adds to the randomness of the approach so I can either merge or obtain a higher obtain a lower number of functional units on transistors or split that means I have or allocate a certain number of temporary registers two are given in the hardware registers it then when I do a split these set of temporary registers will be divided into two sets.

And these two sets will be implemented using two distinct hardware registers you right so in my problem now introduce both emerge and a spring and I am saying that each configuration gives me and has an Associated distinct cost that means that for this configuration with this merging and then this merging of the different operation nodes and that and the different register note I will have an associated costs in terms of the functional units required to allocate them the actual allegations that I have made with respect to the operations on the operations the actual hardware registers that I have used to implement the temporary variables that I have and the number of MUX and DEMUXs that I will require for this.

Now what is the cost of a merge the cost of the merge is a reduction in cost due to change in the number of digester or functional units now when I have when I do a merge what happens the number Of functional units required to implement a set of operations reduce the number of hardware registers required to implement a set of temporary registers reduced because I have done a merge but on the other hand arbitration increases because I have a lower number of registers and functional units I require more MUX and DEMUXs for the arbitration so if there can be a possible increase in MUX / DEMUX cost.

(Refer Slide Time: 08:24)



**A Stochastic Solution Approach**

- *Simulated Annealing* approach
- Motivation - Annealing in metals
  - Heat the solid state metal to a high temperature
  - Cool it down very slowly according to a specific schedule.
  - *If Heating temperature is sufficiently high to ensure random state, and*
  - *Cooling process is slow enough to ensure thermal equilibrium, then*
    - *At the end of the process, the atoms place themselves in a pattern that corresponds to the global minimum energy of a perfect crystal.*
    - Big sized crystals with low defects

VLSI Design, Verification and Test 38

And hence to obtain an overall minimum cost solution let us say in terms of area will not be only a reduction will not be the minimization of the numbers and functional units required it will be a balance between the increase in cost due to MUXs and DEMUXs and the decrease in costs due to the number of registers and functional units reducing so we understand that this is again a complicated NP complete problem and we can apply different techniques to solve NP complete problems optimal strategies as we saw the combinatorial approaches we can have hugest strategies we can also have stochastic strategies.

Now currently in this course we have not studied any stochastic solution approach so for solving this generalized for scheduling problem we will use a stochastic solution approach so one stochastic solution approach is a simulated annealing now the simulated annealing is motivated from the annealing process in metals so what happens in the annealing process in metals we hit a solid state metal to a very high temperature and then cool it down very slowly according to a specific schedule now if the heating is sufficiently high to ensure a random state when it is heated.

So when it is a heater at a very high to our very high temperature it is molecules who can move randomly and we can have a random state and then if the cooling is slow enough to ensure a thermal equilibrium then at the end of the process the atom space themselves in a pattern that corresponds to the global minimum energy of a perfect crystal and for with this we obtain large size crystals with load effects so the basic procedure is to heat the metal to a very high temperature and then cool it down slowly using a thermal schedule and so that after the cooling process is done we get big defect-free crystals.



(Refer Slide Time: 10:43)

## A Stochastic Solution Approach

- *Simulated Annealing* approach
  - A probabilistic method applicable where finding an acceptable local optimum in a fixed amount of time is more important than finding the global optimum
  - An iterative improvement heuristic approach
  - Starts with an arbitrary initial configuration and iterates in an attempt to bring the system to a state with the minimum possible cost.

Simulated Annealing

VLSI Design, Verification and Test 36

Now this approach was used in simulated annealing now simulated annealing is a probabilistic method applicable we are finding an acceptable local optimum in a fixed amount of time is more important than finding the global optimum so simulated annealing approach can be used to obtain good solutions which will not be optimal but can often be near optimal if enough amount of time is allocated to solve it can give you a good solution within a fixed amount of time and hence this strategy is used it is an iterative improvement heuristic approach.

So it starts with an initial solution and the goes over heat iterations to find the best solution that is possible for it within a given time so it starts with an arbitrary initial configuration and peter rates in an attempt to bring the system to a state with the minimum possible cost so therefore what happens what it does is let us say this denotes the state space this denotes the states pace of the problem and this denotes the cost in terms of energy for us energy could be say resource cost area cost then performance cost latency etc...

Right and we have want to obtain say the minimum resource design minimum resource allocation and we do how do you how do we obtain this minimum resource allocation by certain

allocation choices of different resources functional units registers and for that we will obtain a certain number of MUXs and DEMUXs so for a given choice of registers and functional units I will have a certain MUXs cost and therefore I will have an overall cost in terms of area and each point on this on this line who gives a solution in terms of the cost for that configuration so these are different each point in this line corresponds to different configurations but configuration in terms of which functional units has been allocated to which operations which temporary variables have been allocated to which hardware registers right.

So that is a configuration and for that configuration we will have a total functional unit cost we will have a total register Cost and we will have a total multiplexer cost and that will give an estimate of the total area that will be required by the circuit now basically the simulated annealing process initially randomly partially randomly with the probability moves over the entire state space in search of the global optimum so at each point it sometimes find the local optimum and then it again randomly chooses or different parts of the state space and tries to ultimately find the global optimum.

Now it is not guarantee that the simulated annealing process will always find the global optimum however it is it is guaranteed that it will find a local it may find a local minima but almost always it finds a good solution which may be very close to the global minimum or global optimum that we have so it is an iterative improvement heuristic approach and we start with an arbitrary initial configuration and iterates in over the state space in an atom to bring the system to a state with the minimum possible cost let us say in terms of area.

So we will now look deeper into the simulated annealing approach what does it do at each step it considers moving to a neighboring state probabilistically so how can it move to a neighboring state for us it will be a merge or a split so currently in the conflict drop I have a certain merging over the conflict run that means I am I input is a conflict graph and then from that input conflict graph I have moved and obtained at a certain time at an intermediate state a certain merging over the conflict graph that means I have a certain number of operations club together in certain resource instances of registers and functional units okay.

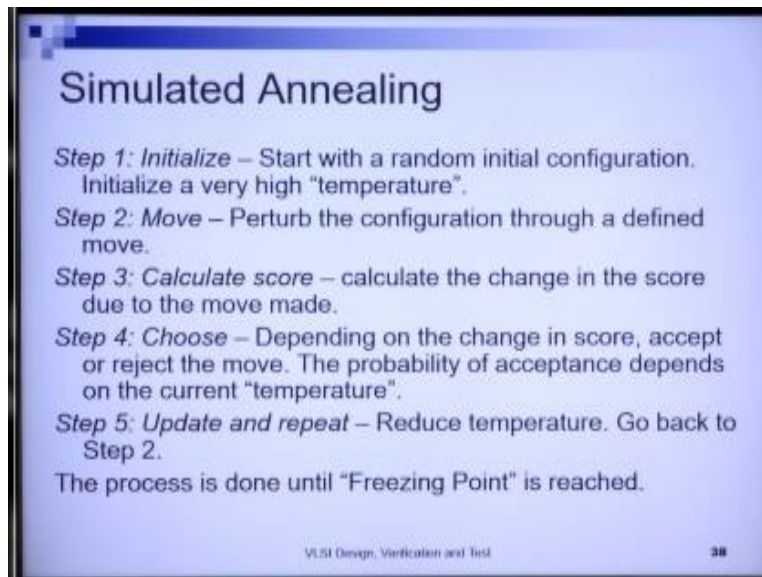
And that gives me an intermediate state and that state has a cost and from there from that state I can move to another state by either merging another register or another functional unit or by splitting a current resist hardware register or functional unit into two different hardware units hardware registers or functional units with two different sets of operation /temporary variables in it so at each step it considers moving to a neighboring state probabilistically moves can be generated either using a merge or a split.

Now what does control schooling means the controlled cooling here in simulated annealing is the slow decrease in the probability of accepting worse solutions as it explores the state space and why do we accept were solutions as or at all accepting word solutions allows for a more extensive search over the state space for example suppose I go on doing emerging and I obtained at this local Optima now if I do further merging here at from this configuration I will not obtain a better solution.

So I have to randomly split / the solution and go to a different part of the state phase and try to minimize from there so but and solution progress is now initially I will take the system to a very random state for exact I can move over the different parts of the state space move over different parts of the state however at later points in time when I am when I am converting to a very good solution I do not want to go out of that of that place and randomly move to a different part of the state space in the in the later iterations.

So initially my probability of accepting were solutions is high so that I can move about in different parts of the state space but as I converse to group good solution after a certain time after a certain number of iterations I want to converge to a certain local minimum which could possibly be a very close to the global minimum that I have so hence I slowly decrease the probability of accepting worse solution as it goes on exploring the solution space.

(Refer Slide Time: 17:57)



So now we look at the steps of the simulated annealing so we start with a random initial configuration fine initialize it to a very high temperature that means I have taken the solution that I have obtained from my heuristic graph coloring algorithm four registers and functional units and use that to obtain a random merging in the initial conflict-free graph we had edges between conflicting operation nodes we had edges between conflicting temporary registers and we had edges between conflicting temporary registers and operations.

Then we generated an initial solution and initial allocation in terms of initial allocation in terms of an allocation of functional you it is and hardware registers to the operations and registers and that gave me a cost of the initial solution in terms of the number of functional units required the area cost of the number of functional units the area cost of the registers and the area cost of the associated MUXs and DEMUXs.

Now from that initial configuration I make moves over the state space at each move as I said I either merge or split with that merger or split where do I go I go to another merging of the of the temporary registers and operations to obtain a different allocation of the functional units and

registers and we will have an associated MUXs and DEMUXs and that moved due to that move from one state to another in the current mean state I will have another cost of the of the solution. So we start with a random initial configuration and initialize it to a very high temperature and then we perturbed the configuration through a defined move so perturbing means I make a move I gotta merge or a split randomly.

So that is what do I mean by per turn then we calculate score we calculate the change in score which is called how for us the score is the cost so the area cost of the solution is our score we calculate the change in area cost or score due to the move made okay now depending on the change in score we either accept or reject the move so the probability of acceptance depends on the current temperature that we have so we said if the current if the move results in an increase in area cost we can still accept the solution in terms of that merge with a certain probability right.

If when the temperature is high but later we will slowly decrease that probability of accepting higher area solutions then we update and repeat we reduce the temperature and go back to step 2 until we need a freezing point.

(Refer Slide Time: 21:09)

**A Stochastic Solution Approach**

- The generic pseudo-code
- $s \leftarrow s_0; e \leftarrow E(s)$  // Initial state, energy.
- $s_{best} \leftarrow s; e_{best} \leftarrow e$  // Initial "best" solution
- $k \leftarrow 0$  // Energy evaluation count.
- while  $k < k_{max}$  and  $e > e_{max}$  // While time left & not good enough:
  - $T \leftarrow \text{temperature}(k/k_{max})$  // Temperature calculation.
  - $s_{new} \leftarrow \text{neighbour}(s)$  // Pick some neighbour.
  - $e_{new} \leftarrow E(s_{new})$  // Compute its energy.
  - if  $P(e, e_{new}, T) > \text{random}()$  then // Should we move to it?
    - $s \leftarrow s_{new}; e \leftarrow e_{new}$  // Yes, change state.
  - if  $e_{new} < e_{best}$  then // Is this a new best?
    - $s_{best} \leftarrow s_{new}; e_{best} \leftarrow e_{new}$  // Save 'new neighbour' to 'best found'.
  - $k \leftarrow k + 1$  // One more evaluation done
- return  $s_{best}$  // Return the best solution found.

VLSI Design, Verification and Test 39

Now we will look at the actual algorithm so initially I have an initial state  $s_0$  and that initial I have an initial states  $0$  I allocate it to  $s$  that initial state and that state is a configuration of initial merging of registers of temporary registers and operations into certain functional into certain hardware registers and functional units that is  $s$  that merging is given by  $s$  and that merging has an associated costs which is given by  $E_s$  so I stored the state and it is corresponding area costs in  $E_s + E$  respectively and this is the initial state and energy.

The energy corresponds before us is the area cost now we all we initially because  $s$  is the only solution that I have is the only merging and he is only cost that I have the current best the current best state then the current best cost is  $s + E$  so I allocate best as best  $s$  based =  $s$  and  $E$  best that is area cost equals to  $E$  so initial best cost solution are a look noted here are given here now we have an energy evaluation count that means basically this is a counter as to how many iterations the algorithm will move through when will I say that I have still not obtain a good enough solution when either the energy cost that I have is still greater than an acceptable  $E_{MAX}$ .

That means the area cost is greater than the maximum acceptable area that I have until this and I have I have still not exhausted the maximum number of iterations if I have already got a solution that is acceptable I can stop the algorithm even before  $k$  has reached  $k_{max}$  right now at any given iteration the temperature is given by temperature  $K / K_{max}$  so  $k_{max}$  is the maximum number of iterations  $k$  is a current number of iterations so  $k / k_{max}$  will initially be a very low value and slowly  $k = k$  will start with  $0$  and slowly it will go it will go higher and higher as  $K$  increases.

And ultimately we want if all if I exhaust all my iterations if I go over all my iterations finally  $k / k_{max}$  will be equal to  $1$  so when  $k / k_{max}$  is  $0$  the temperature is highest so initially I do a temperature in temperature calculation and the temperature is initially very high so when  $K / K_{max}$  is very low the temperature becomes very high now I make a move now after I make a move I get a new solution which is give witches which is taken down in  $s_{new}$  so  $s_{new}$  equals to neighbor of  $s$  so from  $s$  I make a move and that move is either emerged or a split that merge order split gives me a new solution which is given which is given by  $s_{new}$  and I have a corresponding energy costs are an area cost which is given by  $E_{s_{new}}$  right.

Now we accept this new solution we accept this new solution if  $PE_{new}$  is greater than some random value if the probability of accepting this new solution which is given by  $EP(E)$ ,  $U, t E$  is a current solution  $E_{new}$  is the new solution that I have got if this probability is greater than a certain random value I accept this new solution  $s = new$  and  $E = new$  and now  $s$  and  $e$  becomes my new current solutions after this if  $e_{new}$  is less than  $e_{best}$  if the new solution is less than  $m$  then I have a better solution than the current base that I have so I need to update  $e_{best}$  and  $e_{best}$  now.

And then I go to the next iteration so how does this acceptance probability and calculated so  $Pe_{new}$ ,  $T$  is given is equal to 1 if  $u$  is less than so I always accept the new solution if the new solution has a lower cost than the current solution otherwise I will still accept the new solution provided this value is higher than the random value here and what does this value make when  $e_{new}$  is greater than  $e$  that means the new area cost is higher than the current area cost than this value is  $1 / e^{(e_{new} - e) / d}$  so  $1 / e^{(e_{new} - e) / t}$  this value will be higher when  $T$  is very high when  $T$  is very high then what happens when  $T$  is very high then  $1 / e^e$  this value becomes very low when  $e$  to the power this value becomes very low  $1 / e$  to the power this value becomes higher comparatively.

(Refer Slide Time: 27:19)

## A Stochastic Solution Approach

- Acceptance Probability  $[P(e, enew, T)]$ 
  - = 1 if  $enew < e$
  - =  $\exp(-(enew - e) / T)$ , otherwise
- Annealing Schedule  $[T]$ 
  - Initially set to a high value (or infinity)
  - Decreased at each step following an annealing schedule
  - Ends with  $T = 0$  towards the end of the allotted time budget
- How do we choose whether to SPLIT or MERGE?
- One possible approach
  - Initially allow SPLIT and MERGE with equal probability
  - Gradually, increase probability of MERGE

VLSI Design, Verification and Test 40

And then I have a higher chance of this  $P_e$ ,  $E$ ,  $U$ ,  $T$  becoming higher than the random and therefore I have a probability of accepting our solution if this value  $1 - \exp(-\frac{e - enew}{T})$  is greater than the random value.

(Refer Slide Time: 27:34)

## A Stochastic Solution Approach

- The generic pseudo-code

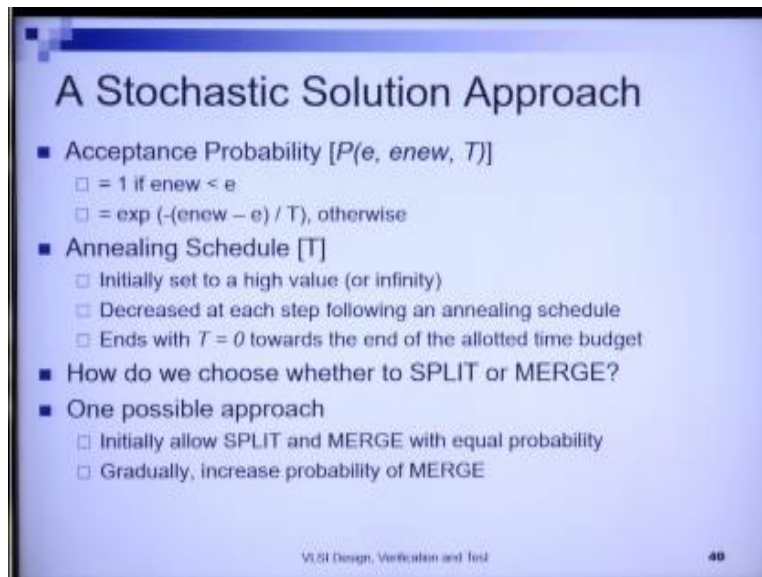
```
■ s ← s0; e ← E(s) // Initial state, energy.
■ sbest ← s; ebest ← e // Initial "best" solution
■ k ← 0 // Energy evaluation count.
■ while k < kmax and e > emax // While time left & not good enough:
  □ T ← temperature(k/kmax) // Temperature calculation.
  □ snew ← neighbour(s) // Pick some neighbour.
  □ enew ← E(snew) // Compute its energy.
  □ if P(e, enew, T) > random() then // Should we move to it?
    ■ s ← snew; e ← enew // Yes, change state.
  □ if enew < ebest then // Is this a new best?
    ■ sbest ← snew; ebest ← enew // Save 'new neighbour' to 'best found'.
  □ k ← k + 1 // One more evaluation done.
■ return sbest // Return the best solution found.
```

VLSI Design, Verification and Test 39



So as we said we always accept a new solution that is better than the current solution we accept a worse solution with a probability and this probability reduces as temperature reduces when the temperature initially is very high then the probability of acceptance of a worse solution is higher than the probability of accepting worse solutions when I slowly reduce the temperature if the new solution is better than the best solution that I have update the best solution and then I go to the next iteration and in this process I go on taking moves until I come down and freeze on to a good solution and after some time as we said the probability of accepting more solution will reduce we will finally down into a local minima and because of the initial randomness in choosing solutions in choosing good and bad solutions I there is a high probability of a boiling down into a solution which gives me which gives me a very good solution may be some times the optimal solution as well.

(Refer Slide Time: 28:56)



**A Stochastic Solution Approach**

- Acceptance Probability [ $P(e, e_{new}, T)$ ]
  - = 1 if  $e_{new} < e$
  - =  $\exp(-e_{new} - e) / T$ , otherwise
- Annealing Schedule [ $T$ ]
  - Initially set to a high value (or infinity)
  - Decreased at each step following an annealing schedule
  - Ends with  $T = 0$  towards the end of the allotted time budget
- How do we choose whether to SPLIT or MERGE?
- One possible approach
  - Initially allow SPLIT and MERGE with equal probability
  - Gradually, increase probability of MERGE

VLSI Design, Verification and Test 40

And how will the annealing schedule 40 that means how will the temperature change initially the temperature as I said is set to a very high value and it and it is decreased at each step following an aniline schedule and it ends at  $t$  equals to 0 towards the end of the alligator time budget right how do we choose whether to merge or split that is also another question one choice is that initially I allow splits and merge with equal probability and then I gradually increase the probability of merges because finally I want to have a more mergers then I split that means finally I want to have a higher resource sharing I need to obtain a resource sharing that gives me the minimum area cost with this we come to the end of this module.

**Centre For Educational Technology  
IIT Guwahati  
Production**

**Head CET  
Prof. Sunil Khijwania**

**CET Production Team  
Bikask Jyoti Nath  
CS Bhaskar Bora  
Dibyajyoti Lahkar**

**Kallal Barua  
Kaushik Kr. Sarma  
Queen Barman  
Rekha Hazarika**

**CET Administrative Team  
Susanta Sarma  
Swapan Debnath**