Embedded Systems – Design Verification and Test

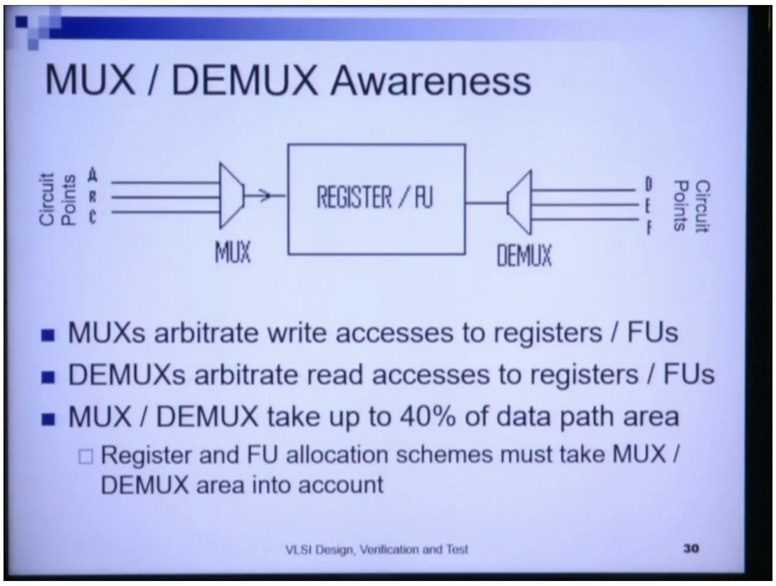
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 11
Hardware Architectural Synthesis – 6

Hello welcome to module 3 of lecture 7. In the last module we looked 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 graph. Such allocation can be optimally done in polynomial time by their corresponding interval conflict graphs 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 mapped to simple interval graphs. And hence the conflict graph coloring problems also become NP complete. And therefore, enumerative techniques are required to solve them. What we used heuristic technique graph heuristic graph coloring techniques to solve that problem; however, in the last lecture we dealt with the allocation and mapping of single resource types. In this lecture we will deal with the generalized resource allocation problems.

(Refer Slide Time: 01:54)


MUX / DEMUX Awareness

Circuit Points
A
B
C
MUX
REGISTER / FU
DEMUX
D
E
F
Circuit Points

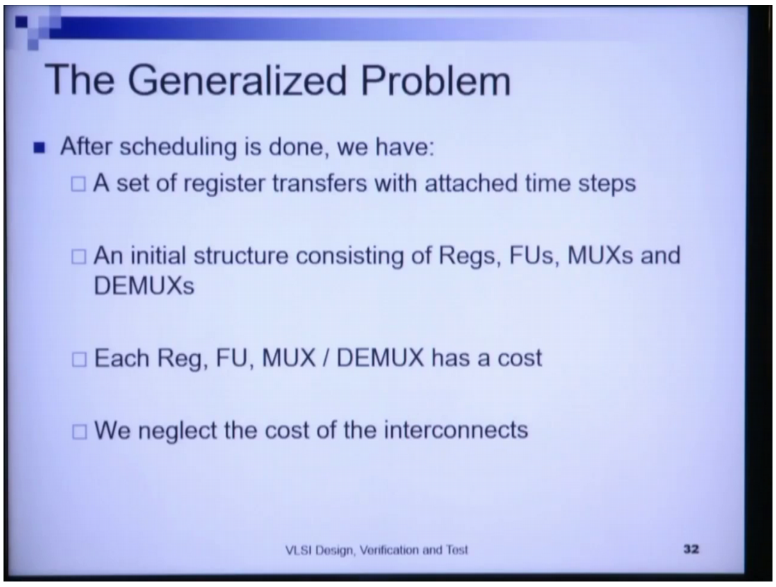
- MUXs arbitrate write accesses to registers / FUs
- DEMUXs arbitrate read accesses to registers / FUs
- MUX / DEMUX take up to 40% of data path area
  - Register and FU allocation schemes must take MUX / DEMUX area into account

VLSI Design, Verification and Test 30

In the introduction to resource allocation and binding, we said that a certain choice of resource allocation effects further dependent resource allocation. For example, we said that register and functional unit allocation choice, will also determine the MUX choice, why? Because the number of the types of functional units that we have as well as registers that we have will determine how many MUXs slash DEMUXs. And what will be their type of MUXs and DEMUXs that we determine by that choice?

And we said that MUXs arbitrate write accesses to registers or functional units. For example, here we have 3 circuit components; circuit points which put their output on the register slash MUX's. And the outputs of this registers and MUXs again go to circuit components def right. And the DEMUXs arbitrate read accesses to register the functional units and MUX DEMUX take up to 40 percent of the data path. And hence their allocation problems cannot be ignored. With this we come to the generalized problem. The generalized post scheduling problem may be posed as follows.

(Refer Slide Time: 03:30)



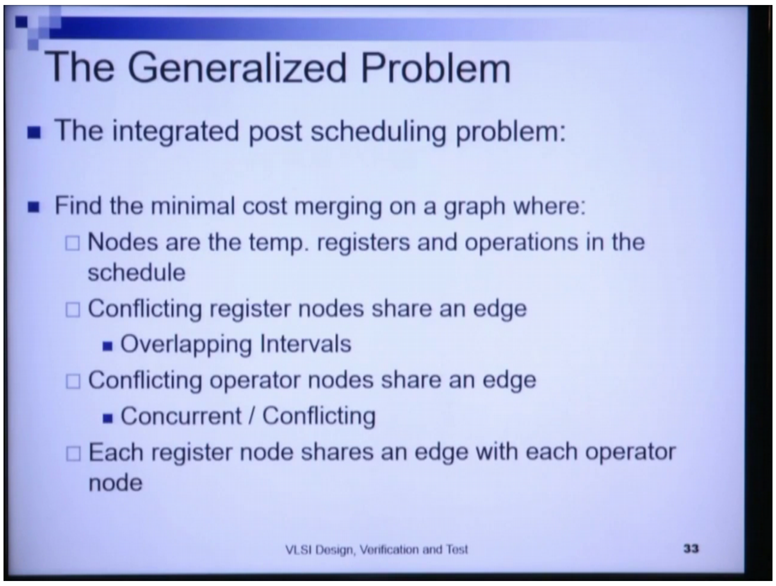
The Generalized Problem

- 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
  - Each Reg, FU, MUX / DEMUX has a cost
  - We neglect the cost of the interconnects

VLSI Design, Verification and Test 32

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 same; that means, we have done an initial register, allocation functional unit allocation MUX and DEMUXs allocation. By say, our heuristic graph coloring algorithm that we studied in the last module. And each register for each register functional unit MUX and DEMUX we know it is cost, and we currently neglect the cost of interconnects.

(Refer Slide Time: 04:02)



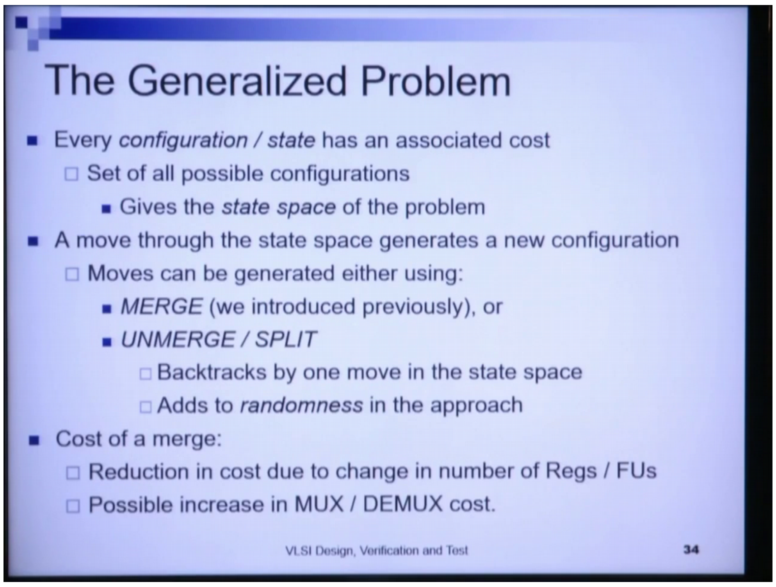
The Generalized Problem

- 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

So, given this scenario the integrated post scheduling problem is, to find a minimal cost merging on the conflict graph, where nodes are the temporary registers and (Refer Time: 04:14) operations in the schedule, conflicting register nodes share an edge conflicting operation nodes. This is not operator, but operation conflicting operation nodes also share an edge. And each register nodes shares an edge with each register with each operation, right.

(Refer Slide Time: 04:34)



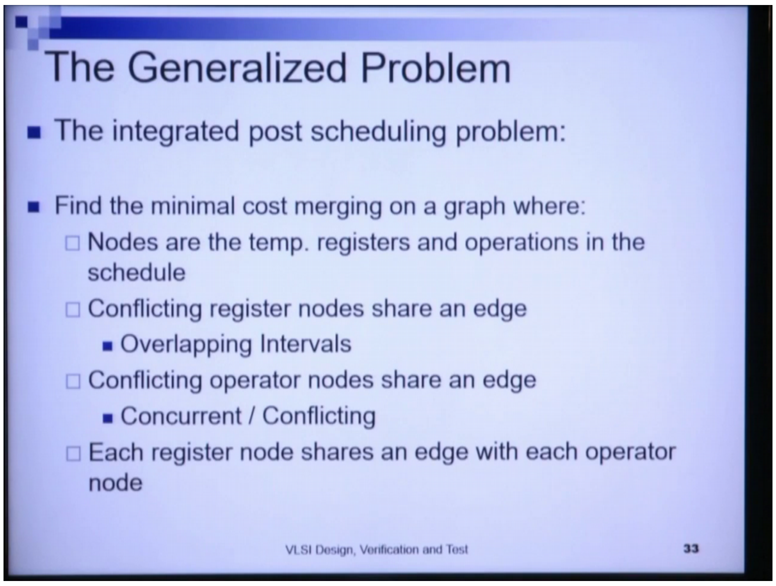
The Generalized Problem

- 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

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 MUXs DEMUXs.

(Refer Slide Time: 04:56)


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

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 total number of registers required for allocation total number of MUXs and DEMUXs required for allocation. And that gives the cost of the configuration, right. Set of all possible configurations gives the state space of the problem.

Now what the different possible types of mergings that we can do on this conflict graph, gives me the different possible choices that I have and gives the total state space of the problem. A move through the state space generates a new configuration. A move through the state space generates a new configuration. And moves can be generated using either a MERGE which is previously introduced.

So, when we have a MERGE, what do we do? We MERGE 2 registers together; that means, that I am allocating the same register instance to 2 temporary registers. I am merging 2 operations together meaning I am allocating the same functional unit instance to these 2 operation. So, we MERGE we introduced previously, now we introduce a SPLIT a SPLIT backtracks 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 or registers or SPLIT; that means, I have allocate certain number of temporary registers to a given hardware registers. Then when I do a SPLIT these set of temporary registers will be divided in to 2 sets. And these 2 sets will be implemented using 2 distinct hardware registers. You write, so in my problem now I introduce both the MERGE and the SPLIT and I am saying that each configuration gives me an has an associated distinct cost. That is that for this configuration with this merging, then this merging of the different operation nodes and that and the different register nodes.

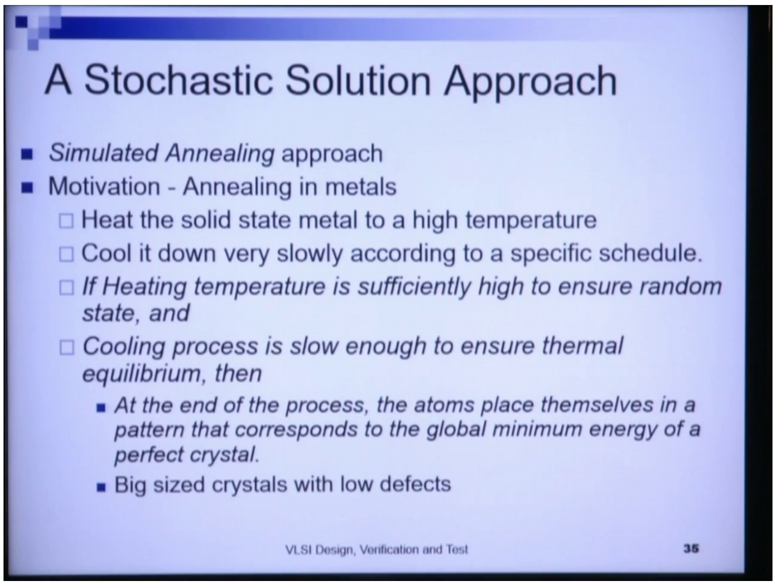
I will have an associated cost in terms of the functional units required to allocate, then the actual allocations that I have made with respect to that operation on the operation the actual hardware registers that have used to implement the temporary variables that I have, and the number of MUXs and DEMUXs that I will require for this. Now what is the cost of a MERGE? The cost of a MERGE is a reduction in cost due to change in the number of registers of functional units.

Now, when I have, when I do MERGE what happens? A number of functional units required to implement a set of operations reduced then 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 lower number of registers and functional units, I require more MUXs DEMUXs for the arbitration.

So, it there can be a possible to increase in MUX like DEMUX cost. 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 we require. It will be a balance between the increase in cost due to MUXs and DEMUXs, and the decrease in cost 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 the NP complete problem. Optimal strategies as we saw the combinatorial approaches. We can have heuristic strategies we can also have stochastic strategies. Now currently in this course, we have not studied any stochastic solution approach.

(Refer Slide Time: 09:22)



A Stochastic Solution Approach

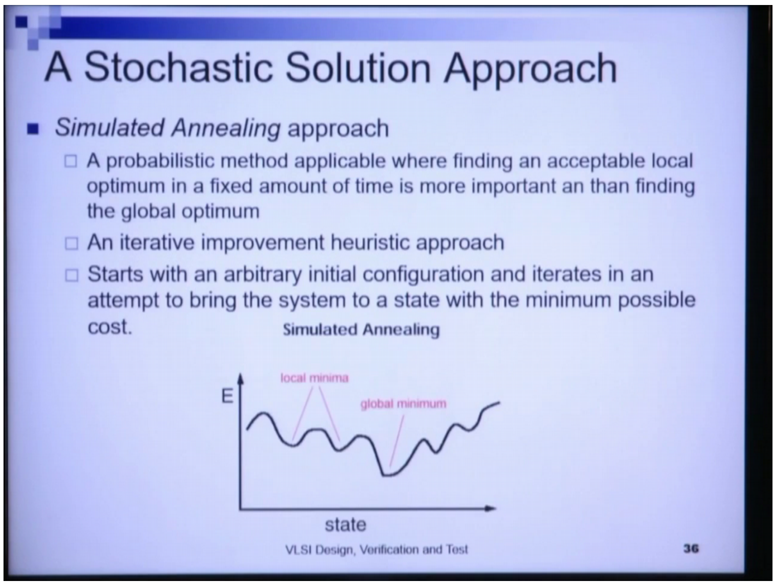
- *Simulated Annealing* approach
- Motivation - Annealing in metals
  - Heat the solid state metal to a high temperature
  - Cool it down very slowly according to a specific schedule.
  - *If Heating temperature is sufficiently high to ensure random state, and*
  - *Cooling process is slow enough to 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 35

So, for solving this generalized post 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 heat 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 heated at a very high to a head very high temperature it is modelling molecules can move randomly and we can have a random state. And then if the cooling is slow enough ensure a 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. And for with this we obtain large sized crystals with low defects. 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: 10:48)



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 an 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.

VLSI Design, Verification and Test 36

Now, this approach was used in simulated annealing. Now simulated annealing is a probabilistic method applicable where 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.

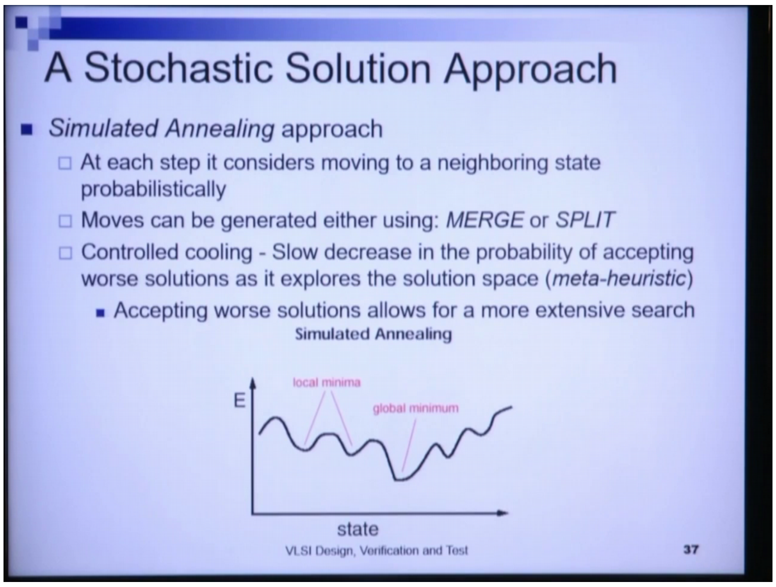
It can give you a better solution within a fixed amount of time and hence this strategy is used. It is a iterative improvement heuristic approach. So, it starts with an initial solution and goes over iterations to find the best solution that is possible for it within a given time. So, it starts with an arbitrary initial configuration and iterates 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 state space of the problem. And this denotes the cost in terms of energy.

For us it could energy could be say resource cost area cost then performance cost latency etcetera, right and we are one to obtain say the minimum resource design minimum resource allocation. And we how do you how do we obtain this minimum resource allocation? By certain allocation choices of differ different resources functional units registers and for that we will obtain a certain number of MUXs DEMUXs.

So, for a given choice of registers and functional units I will have a certain MUX cost and therefore, I will have an overall cost in terms of area. And each point on this on this line gives a solution in terms of the cost for that configuration. So, these are different co each point in this line corresponds to different configurations. 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 a different parts of the state space and tries to ultimately find the global optimum. Now it is not guaranteed that simulated annealing process will always find a global optimum; however, it is guaranteed that it will find local it may find a local minima, but almost always it finds the good solution which may be very close to the global minimum or global optimum that we have. So, it is a iterative improvement heuristic approach and it start with an arbitrary initial configuration and iterates in over the state space, in an attempt to bring the system to the state with the minimum possible cost, ok. Let us see in terms of area.

(Refer Slide Time: 14:37)



A Stochastic Solution Approach

- *Simulated Annealing* approach
  - At each step it considers moving to a neighboring state probabilistically
  - Moves can be generated either using: *MERGE* or *SPLIT*
  - Controlled cooling - Slow decrease in the probability of accepting worse solutions as it explores the solution space (*meta-heuristic*)
    - Accepting worse solutions allows for a more extensive search

VLSI Design, Verification and Test 37

So, will now look deeper in to the simulated annealing approach, what does it do at each step it considers moving towards the 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 graph, I have a certain merging over the conflict graph; that means, I am my input is a conflict graph, and then from that input conflict graph, I have moved and obtained at a certain time at a intermediate states a certain merging over the conflict graph; that means, I have a certain number of operations kept together in certain resource instances of registers and functional units, ok.

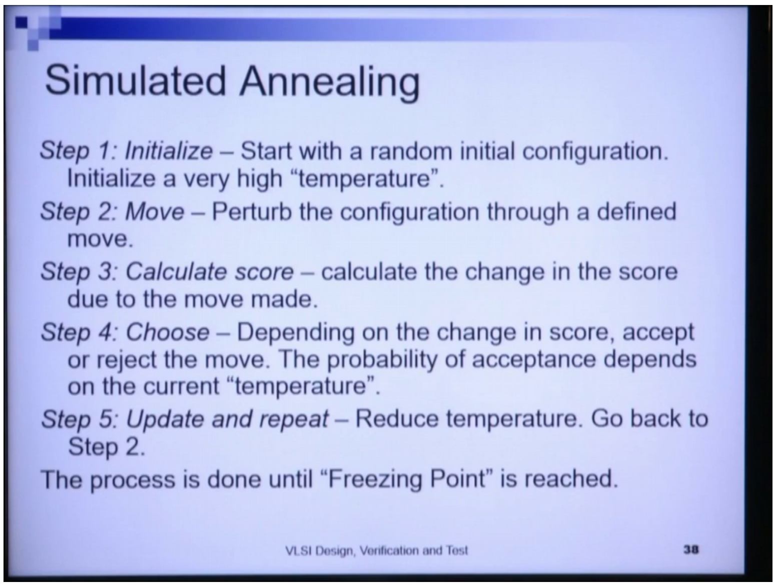
And that gives me an intermediate states, and that state has a cost and from that from that state I can move to another state by either merging another register or another functional unit or by splitting a current hardware register or functional unit into two different hardware units hardware registers or functional units two different sets of operation slash 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 controlled cooling 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 worse solutions as at all? Accepting worse solution allows for a more extensive search over the state space. For example, suppose I go on doing a merging and I obtain at these 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 perturb the solution and go to a different part of the state space and try to minimize from there; so, but as a solution progresses. Now initially I will take the system to a very random state for ex. So, that I can move over the over different parts of the state space move over the different parts of the state space, but; however, when at later points in time when I am when I am converging 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 this state space; in a in the in the later iterations.

So, initially my probability of accepting worse solutions is high. So, that I can move about in different parts of the state space, but as I converge to move good solutions 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 to a solution as it goes on exploding the solution space.

(Refer Slide Time: 18:00)



Simulated Annealing

Step 1: Initialize – Start with a random initial configuration. Initialize a very high "temperature".

Step 2: Move – Perturb the configuration through a defined move.

Step 3: Calculate score – calculate the change in the score due to the move made.

Step 4: Choose – Depending on the change in score, accept or reject the move. The probability of acceptance depends on the current "temperature".

Step 5: Update and repeat – Reduce temperature. Go back to Step 2.

The process is done until "Freezing Point" is reached.

VLSI Design, Verification and Test 38

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 for registers and functional units and use that to obtain a random merging. In the initial conflict 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.

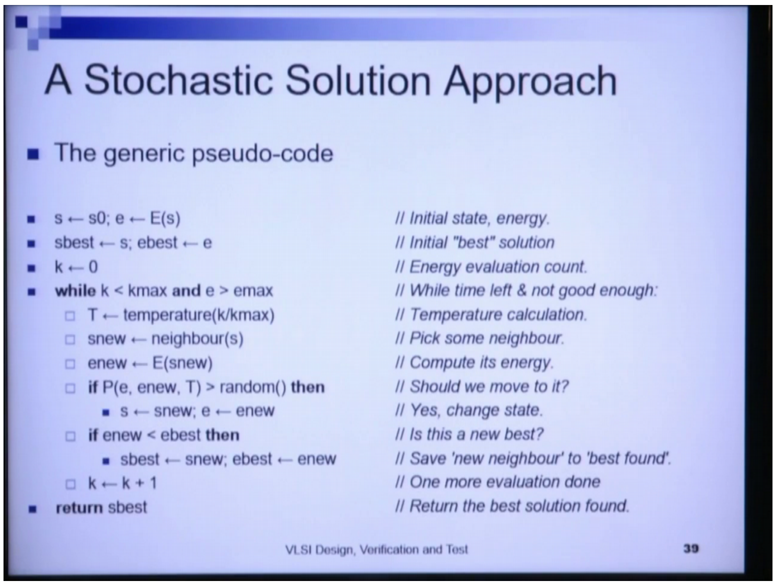
Then we generated an initial solution an in initial allocation in terms of initial allocation, in terms of an allocation of functions units and hardware registers to the operations and registers. And that gave me a cost of the initial solution in terms of the number of functional units required, the area cost of the number of functional units the area cost of the register 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 MERGE or SPLIT. With that MERGE or SPLIT where do I go I go to another merging of the of the temporary registers and operations to obtain different allocation of functional units and in registers and we will have an associated MUX and DEMUXs cost and that move due to the that move from one state to another in the current I will have another cost of the of the solution. So, we start with a random initial configuration and initialized it to a very high temperature. And then we perturb the configuration through a defined move.

So, perturbing means I make a move either a MERGE or a SPLIT randomly. So, that is what do I mean by perturb. Then we calculate score calculate the change in score which score for us the score is the cost. So, the cost area cost of the solution is of our score, we calculate the change in area cost or score due to the move made. Now, depending on the change in score we either accept or deduct 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 a increasing 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 a higher area solutions. Then we update a repeat we reduce the temperature and go back to step 2 until we reach a freezing point.

(Refer Slide Time: 21:11)



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

Now, we will look at the actual algorithm. So, initially I have an initial state s 0, and that initial I have an initial state s 0. I allocate it to s, that initial state, and that state is a configuration of initial mergings of register of temporary registers and operations in to certain function in to certain hardware registers and functional units; that is s that merging is given by s. And that that merging has an associated cost which is given by E

s. So, I store the state and it is corresponding area cost in s and e respectively. And this is the initial state and energy. The energy for us is the area cost. Now we all we initially, because s is the only solution that I have is only merging and is only cost that I have, the current best the current best state and the current best cost is s and e.

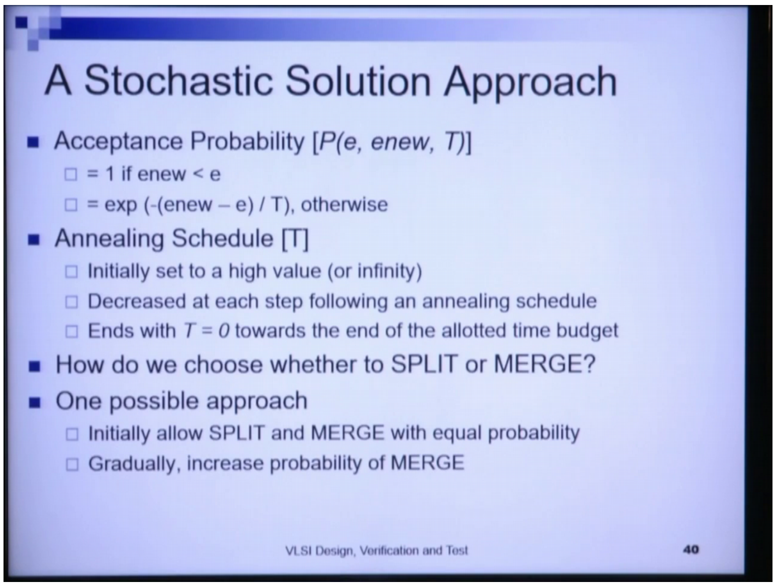
So, I allocated best sbest equals to s and ebest that is area cost equal to e. Initial best cost solution are noted here, are given here. Now we have an energy evaluation count; that means, basically this is the counter as to how many iterations the algorithm will move through. When will I say that I have still not obtained a good enough solution, when either the energy cost that I have is still greater than an acceptable max. That means, the area cost is greater than the maximum acceptable area that I have, the until this happens and 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 the temperature k by k max.

So, k max is the maximum number of iterations, k is the current number of iterations. So, k by k max will initially be a very low value. And slowly k equals to k will start with 0, and slowly it will the 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 iterations finally, k by k max will be equal to 1. So, when k by k max is 0 the temperature is highest. So, initially I do what temper temperature calculation and the temperature is initially very high. So, when k by 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 g which is which is taken down in snew.

So, snew is equals to neighbor of s. So, from s I make a move, and that move is either a MERGE or a SPLIT. That MERGE or a SPLIT gives me a new solution which is which is given by snew. And I have a corresponding energy cost or an area cost which is given by e snew, right. Now we accept this new solution, we accept this new solution if P e comma enew comma T is greater than some random value. If the probability of accepting this new solution which is given by e P of e comma enew comma T is a current solution; enew is the new solution that have got.

If this probability is greater than a certain random value I accept this new solution. S equals to snew and e equals to enew and s and e becomes my new current solutions after this. If enew is less than ebest, if the new solution is less than ebest, then I have a better solution than the current best that I have. So, I need to update ebest and e sbest and ebest now. And then I go to the next iteration.

(Refer Slide Time: 25:54)



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

So, how does this acceptance probability calculated? So, P e comma enew comma T is given is equal to 1 if enew is less than e. So, I always accept the new solution, if the new solution has a lower cost than the current solution. Otherwise I may still accept the new solution provided this value is higher than the random value here. And what does this value make? When enew is greater than e; that means, the new area cost is higher than the current area cost, then this value is 1 by e to the power enew minus e by t. So, 1 by e to the power enew minus e by t this value will be higher when t is very high.

So, when t is very high then what happens? When t is very high then 1 by e to the power e to the power this value becomes very low when e to the power this value becomes very low 1 by e to the power this value becomes high comparatively higher and then I have a higher chance of this p e comma enew comma T becoming higher than the random. And therefore, I have a probability of accepting a worse solution if this value is 1 by e to the power enew minus e by t is greater than the random value.

So, as we said we always accept the new solution that is better than the current solution. We accept a worse solution with a probability. This probability reduces as temperature reduces when the temperature initially is very high, then the probability of acceptance is a worse solution is higher than the probability of the accepting worse solutions when I used to only reduce the temperature, if the new solution is better than the best solution that I have I 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 sometime as we check the probability of accepting worse solution will reduce.

We will finally, boil down in to local minima. And because of the initial randomness in choosing solutions, in choosing good and bad solutions, I there is a higher probability of boiling down into a solution which gives me a good solution may be sometimes a optimal solution as well. And how in the annealing schedule for T that is how in the temperature change? Initially the temperature change as I said in is said to a very high value, and it is decreased at each step following an annealing schedule.

And it ends at t equals to 0 towards the end of the allocated 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 more MERGEs then I SPLIT; that means, finally, I want to have higher resource sharing. I need to o obtain a resource sharing that gives me the minimum area cost. With this we come to the end of this module.