

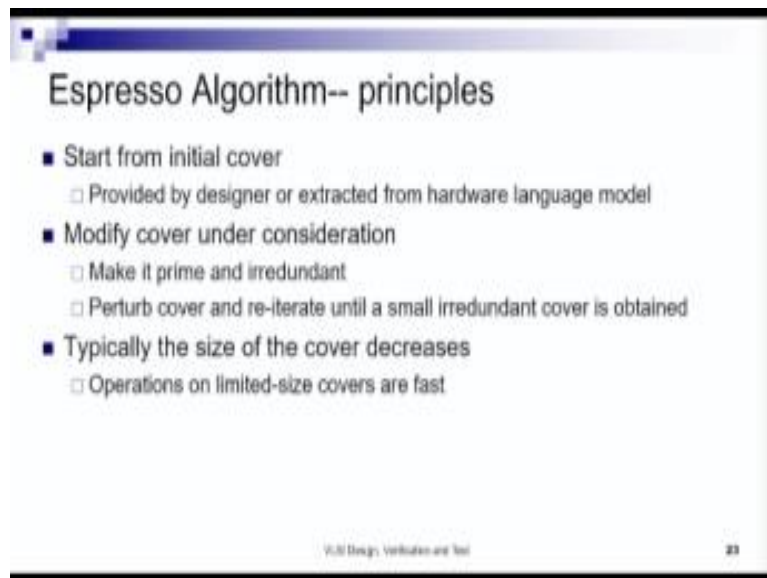
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATHI
NPTEL
NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

VLSI Design, Verification & Test

Dr. Arnab Sarkar
Dept. of CSE
IIT Guwahati

Hello now we discuss the espresso algorithm so what does espresso do?

(Refer Slide Time: 00: 32)



It is a heuristic strategy it starts with an initial cover which is provided by the designer so initial cover over the set of product terms which over the set of min terms over the set of implicants right and with this initial cover it is a cover because it covers at all the one terms in the function right all the one product terms in the function is somehow covered by this initial cover.

So provided by designer or extracted from Hardware language model right then we modify the cover under consideration how do we do it we do it essentially through a series of expansions to form prime implicants and then cover the set of once minimally right and find the minimal cover from the prime implicants that we have found then we do reduction and

then again expand in another direction so this is the basic principle which is followed in the espresso algorithm.

So given an expression it tries to see which terms can be reduced literally and in you reduce literally means we can reuse the number of literals in it to form primes and then we determine its cover reduce it and expand it another to add expand in another direction this is repeated until a suitable solution is reached so basically what do we do we make it right by expanding we make it prime and is redundant.

And that means then we find all the prime implicants and then we discard all the redundant prime implicants we find a cover over primes which is not redundant with meaning that if we remove another prime implicant it will un cover some of the ones not being covered by any other prime implicant right so this is an redundant cover after expanding all the all the ones to the prime implicants we form get and we obtain an e redundant cover over the function and we then perturb the solution by this reduction step perturbs.

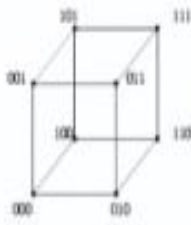
this is a solution using a set of prime implicants which cannot be further reduced because in the if we had expanded in this way then it cannot be further reduced so therefore to now obtain a better solution what do we do is to perturb the solution through a set of reductions we will talk to have talked about all these steps in detail and then reiterate until a small redundant cover is obtained so we often obtain good solutions however.

The solution in most cases will not be optimal this is a heuristic methodology and typically the size of the cover reduces and we over time and we obtain a limited size cover now before discussing the espresso algorithm in detail we need to understand how are how is the function represented in the espresso algorithm so it is represented inside a computer using a data structure whose abstract representation I will present here so the mean terms in the function are represented on the vertices of a hypercube right.

(Refer Slide Time: 04:14)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

©2003 Design Verification and Test

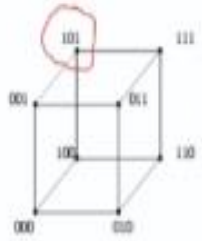
24

For example here this is a min term this is another midterm.

(Refer Slide Time: 04: 21)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

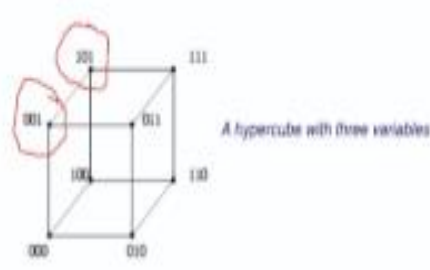
VLSI Design: Verification and Test 24

This is another minterm.

(Refer Slide Time: 04:22)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

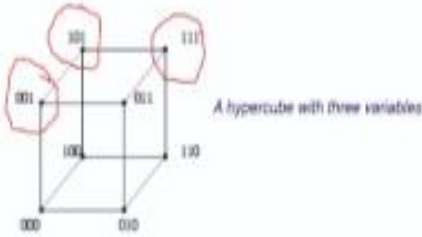
VLSI Design, Verification and Test 24

So this is a three variable function.

(Refer Slide Time: 04:26)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

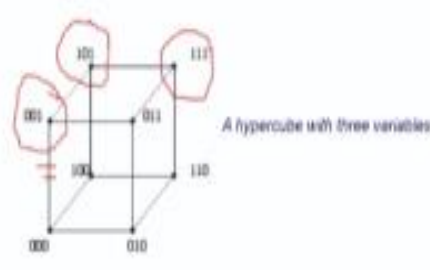
VLSI Design: Verification and Test 24

And all these min terms all the eight min terms have been represented on the vertices of a hyper cube here this is a cube or three dimension on every edge connecting two vertices for example this edge.

(Refer Slide Time: 04:43)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

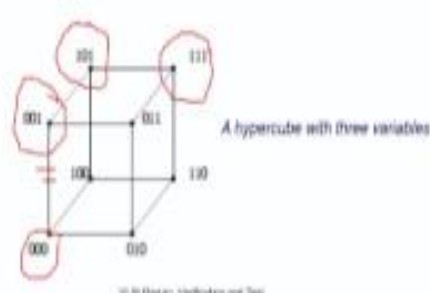
VLSI Design, Verification and Test 24

Here every edge connecting to what is a stay only one bit position change in only one bit position between adjacent min terms so this one is a min term here this one is a min term.

(Refer Slide Time: 04: 55)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

© 2000 Design, Verification and Test

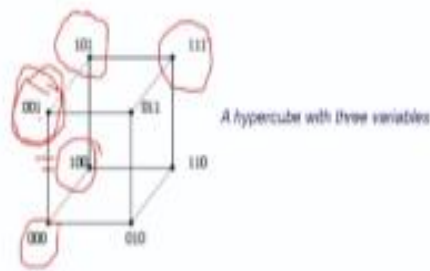
24

Here this min term is 0 0 0 this min term is 0 0 1 so it has changed in only one position the LSB position similarly if you see this one is 0 0 0 this one is 1 0 0.

(Refer Slide Time: 05:13)

Espresso – Abstract Representation

- Minterms are represented on the vertices of a hypercube
- For every edge connecting two vertices, change in only one bit position between adjacent minterms.



A hypercube with three variables

VLSI Design: Verification and Test 24

It is again change in only 1 1 position so adjacent min terms which are connected by an edge on the cube always change by one bit position always change in one bit position.

(Refer Slide Time: 05: 27)

Espresso – Abstract Representation

- For four variables two 3-D hypercubes are made and connected accordingly

- For five variables, duplicate this and connect accordingly

VLSI Design: An Introduction and Test 25

Now from the three variable representation how do we represent four variables for four variables we first obtain two three variable hypercube.

(Refer Slide Time: 05:39)

Espresso – Abstract Representation

- For four variables two 3-D hypercubes are made and connected accordingly

- For five variables, duplicate this and connect accordingly

VLSI Design: Verilog and Test

25

These are to say in three variable hypercube and then we make connections accordingly for example this is a connection.

(Refer Slide Time: 05:50)

Espresso – Abstract Representation

- For four variables two 3-D hypercubes are made and connected accordingly

- For five variables, duplicate this and connect accordingly

VLSI Design, Verification and Test 29

So between this and this we see that only the MSB changes right.

(Refer Slide Time: 05:50)

Espresso – Abstract Representation

- For four variables two 3-D hypercubes are made and connected accordingly

- For five variables, duplicate this and connect accordingly

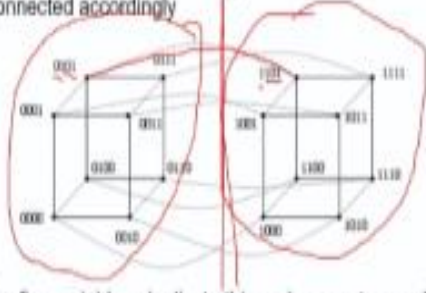
VLSI Design, Verification and Test 25

In all other respects this three variable hyper cube and this three variable hypercube is same.

(Refer Slide Time: 05:59)

Espresso – Abstract Representation

- For four variables two 3-D hypercubes are made and connected accordingly

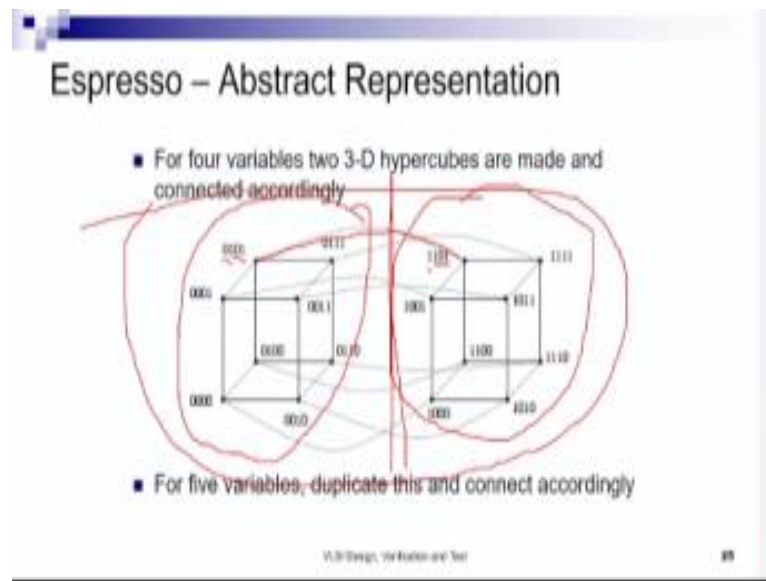


- For five variables, duplicate this and connect accordingly

VLSI Design: Verification and Test 29

For five variables we duplicate this whole structure this whole structure together.

(Refer Slide Time: 06: 04)



We duplicate this and again make connections accordingly now before looking at the espresso algorithm in detail we first go and look at each of its function.

(Refer Slide Time: 06: 16)

Espresso Algorithm - Expand

- **expand (F, R)**
 - It tries to expand the cubes in F with neighboring cubes and with nodes in the DC-set to form larger cubes
 - It takes essential sub-cubes and tries to expand them till they become prime sub-cubes.
 - An essential cube is a cube containing a minterm covered by no other cube.
 - A prime cube is a cube which is fully expanded against the OFF-set and cannot be expanded further.

VLSI Design, Verilog and Test

27

So we said that most important steps in the espresso algorithm is a loop is a loop which goes on reducing expanding and getting an e redundant cover this happens over and over again until the cost is stabilized so these three functions expand reduce and even under these are the three most important functions in the espresso algorithm and therefore we need to discuss these functions first before understanding the algorithm in detail.

So what does the expand function loop expand function takes as input f & R where f represents the current covered over implicants that we have to get the onset of the function and we also have are the set of do not cares what does expand try to do it tries to expand the cubes in F so we said that we first initially we have a random initial cover from this random initial cover.

So this random initial cover will consist of a set of cubes will consist of a set of coverings which we call cubes or sub cubes from the entire cube representation that we have so it tries to expand the cubes in f with neighboring cubes and with nodes in the DC set to form larger cubes so what does it try to do it expands cubes in F so it takes one Cuban s and tries to expand in a certain direction until no further expansion is possible and how can it expands by using either one terms or do not care terms in a given direction.

So when no no further expansion on one direction is possible then it will try to expand in another direction now how does it try to expand it takes essential sub-q band tries to expand them till they become prime sub cubes what are essential sub tubes and essential sub-q busy tube containing a min term covered by no other cube it takes an essential sub-q bin f and expands it until it is prime.

This is what we said a prime to visit cube which is fully expanded against the offset and cannot be expanded further if you further try to expand the prime cube by reducing one more literal say in it then it will cover some part of the offset okay so therefore it cannot be expanded as an example let us say this was an initial cube in F.

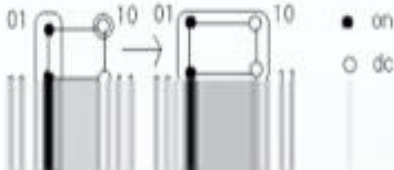
This was an initial cube minus and these are two do not cares okay these are two do not leaves then we can expand in this direction and finally we can obtain this cube after expansion okay this cube will be obtained after expansion and we cannot expand it it further because let us say in the vicinity we have zeros somewhere we have zeros and hence we cannot expand this further now by this we can reduce the number of literals in the cube by one okay.

So this is the objective now implicants covered by an expanded implicant are removed from further consideration now we have already removed then it is removed from further consideration expansion reduces the number of terms in the expression okay this is how what we said it reduces the number of terms in the expression why because we are we are taking two or more product.

(Refer Slide Time: 10:18)

Espresso Algorithm - Expand

- **expand (F, R)**
 - Implicants covered by an expanded implicant are removed from further consideration
 - Expansion reduces the no. of terms in the expression ✓
 - R is required so that the function knows which nodes are in the OFFset and which cannot be extended against.
 - Quality of result depends on order of implicant expansion
 - Heuristic methods used to determine order



Combining them into a single product term that is why it reduces the number of terms are required that is the do not care said his set is required. So that the function knows which nodes are in the offset which nodes are in the oxide and which cannot be expanded against because we cannot cover the offset right while expansion.

The quality of the result depends on the order of implicant expansion this is also an important point suppose we have a cube if we expand in Direction X it will give it may give me a much bigger tube than we expand then if we expand in a direction Y so therefore there are loops in this expand reduced in ricks expand ir redundant and introduced this loop tries to expand cubes in different directions and try to obtain a minimal cover over price right.

(Refer Slide Time: 11:19)

Espresso Algorithm - Expand

- **expand (F, R)**
 - Implicants covered by an expanded implicant are removed from further consideration
 - Expansion reduces the no. of terms in the expression ✓
 - R is required so that the function knows which nodes are in the OFFset and which cannot be extended against.
 - Quality of result depends on order of implicant expansion ✓
 - Heuristic methods used to determine order

● on
○ dc

VLSI Design: Heuristics and Iter

28

Over essential primes now heuristic methods are applied determined to to determine this order heuristic methods are applied to determine this order in which order should the implicants be expanded and also in which direction right after this expansion step we arrived at the ir redundant step what does ir redundant step do it determines the minimal set of e redundant cubes covering the onset.

(Refer Slide Time: 11: 55)

Espresso Algorithm - Irredundant

- **irredundant (F, D)**
 - It determines the minimal set of irredundant cubes covering the ON-set.
 - Irredundant Cover
 - no proper subset is also a cover
 - This cover is extracted from the expanded primes

Legend: ● on, ○ dc

VLSI Design: Verification and Test 29

Now we have already expanded using a set of primes but all primes may not be well all primes may not be essential the a few of the prize will be redundant what do we mean by that all the ones in that in that prime cube is covered by some other cubes there is not at least 11 that there is not even 11 within this prime cover that is not covered by any other prime cube and hence this is a and hence this is this is redundant next we come to the reduced step.

(Refer Slide Time: 12: 38)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.

VLSI Design: Verilog and Test

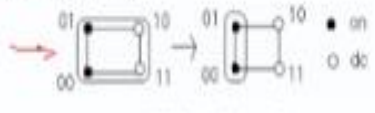
So after we have done the expansion and the redundant steps the solution is usually pretty good but as we said if we had expanded possibly in a different direction and the order in which we had we had decided to expand the implicants the order of the implicants chosen also determined the quality of the cover that we get and hence often after we have we have obtained one round and we have obtained one solution the solution can still be still be made better okay there might exist another cover with fewer min terms and fewer literals this is what we said so what do we do we shrink prime implicants to the smallest size that still covers the onset okay we shrink prime implicants to the smallest size.

That still so then now these curve these a prime implicants will not more not any more remain prime because we are shrinking it however while shrinking we are keeping into consideration that you will not uncover any anybody in the onset we will still keep a cover of everybody in the onset we can reduce the primes but while reducing the primes who suppose I suppose he suppose in this example.

(Refer Slide Time: 14: 14)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.



VLSI Design, Verification and Test

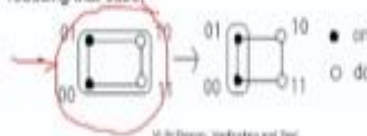
30

I had this cover here and it was a prime cover.

(Refer Slide Time: 14: 17)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.



VLSI Design: Verilog and Test

39

Now when I reduce it.

(Refer Slide Time: 14: 22)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.

VLSI Design, Verification and Test

We still see that here was here was a covered.

(Refer Slide Time: 14: 26)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.

Legend:
● on
○ dc

© 2000 Design, Verification and Test

Because these are do not cares this is still a cover okay however let us say this one is also an on this is not a DC variable as is as is this one now when this was a cover then if I if I reduce it to this if I reduce it to this then this will not be this.

(Refer Slide Time: 15:06)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.

VLSI Design, Verification and Test

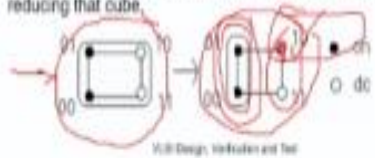
38

This on variable this on vertex will not be covered any more if this one vertex is not covered by another prime implicant from somewhere else like this one say let us say you have another side do you have another cover for this for this on set for this vertex on the onset you do not have another cover then if you reduce it and reduce it to only these two these two vertices you cover only these two vertices.

(Refer Slide Time: 15: 17)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.



And you uncovered this then what happens it will completely not be covered by any other prime implicant and this is not allowed so we shrink prime implicants so what we say is that we shrink prime implicants to the smallest size that still covers the onset this allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained now after I have reduced this I can now possibly expand in a different direction.

To allow this I do the reduction step thus new prime implicants can possibly be obtained by exploring in the new directions the do not cares why have you used the do not cares.

(Refer Slide Time: 16: 11)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - □ Shrink prime implicants to smallest size that still covers ON-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - □ The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.

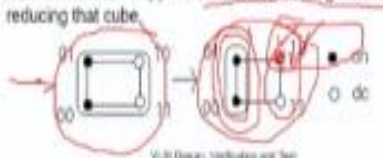
VLSI Design: Verilog and Test 39

Do not cares are required to know which nodes in the cube are do not care and can be dropped without the least risk of losing the cover.

(Refer Slide Time: 16: 18)

Espresso Algorithm - Reduce

- **reduce (F, D)**
 - Solution usually pretty good, but sometimes can be improved
 - Might exist another cover with fewer terms or fewer literals
 - Shrink prime implicants to smallest size that still covers CN-set
 - This allows the newly formed cube to expand in a direction different from the larger cube from which it was obtained.
 - Thus new larger cubes can possibly be obtained by exploring the new directions.
 - The don't care is required to know which nodes in a cube are don't care and can be dropped without risk of losing the cover, while reducing that cube.



While reducing that kill for example here we had to do not cares and therefore we could reduce the cover to this we could reduce the cover to this however we said that if this was not a do not care and the and it was also not covered by any other prime implicant then we could not have done this reduction in this way and this is what the reduction step is now after we have done.

There use expand key redundant steps for a certain number of time so after reduction we will do again an expansion again obtain any redundant cover again reduced and again possibly expand in another different direction after this we and all the while we are on doing this we are keeping the best solution that is obtained till now the minimum cover that is obtained till now the best solution is always kept.

So until the cost stabilizes we will go on doing it after this is done we will take our simulated annealing like approach perturb the solution completely and try to try to go to a different part of the state space and again try to do this reduce expanded redundant step in that part of the states pace now how do we do that perturbation of the current solution.

(Refer Slide Time: 17: 46)

Espresso Algorithm – Expand Gasp

- **expand_gasp(G , R)**
- The function takes as input the set G produced after Reduce Gasp and R.
- It tries to expand cubes and adds them if they cover other cubes.
- R is used to ensure that nodes in the OFF-set are not included.
- The redundant cubes introduced are removed later.

01 11
00 10

01 11
00 10

● on
⊗ off

VLSI Design, Verification and Test 91

We do it us expand you we do it using reduced gasp and expand gasp so first we understand the reduced gasp.

(Refer Slide Time: 17: 18)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 11 00 10 → 01 11 00 10

● on
⊗ off

VLSI Design, Verification and Test 18

So for each cube in F add those sub cubes in F that are not covered by other cubes so what do we have in FS set that current solution is present in the current solution is present in it now the current solution is a current set of implicants which cover the onset of the function now we take that solution the current solution and then add those sub cubes in F that are not covered by other cubes for example here for example here.

(Refer Slide Time: 18: 32)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 11 00 10

01 11 00 10

● on
⊗ off

VLSI Design, Verification and Test 92

We have 40 for each cube in F at those sub cubes in if that are not covered in other two so what do we do here we have two covers and both of them are in F okay.

(Refer Slide Time: 18:50)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 ● N
00 ● 10

01 ● 11 ● on
00 ● 10 ✕ off

VLSI Design, Verification and Test 28

Then we add this one this one is not.

(Refer Slide Time: 18:54)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 11
00 10

01 11
00 10

● on
⊗ off

VLSI Design: Verilog and Test

38

So this one is a node this one is a node.

(Refer Slide Time: 18:58)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 11
00 10

01 11
00 10

● on
⊗ off

VLSI Design: Verification and Test 22

That is covered by both the cubes this cube and this tube ever this cube and this cube is only covered by one sub Q right so we add this disk this cube and this cube to F.

(Refer Slide Time: 19:01)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

The diagram shows a 2x2 grid of nodes. The top-left node is labeled '01' and has a solid dot (on). The top-right node is labeled '11' and has a solid dot (on). The bottom-left node is labeled '00' and has a solid dot (on). The bottom-right node is labeled '10' and has a crossed dot (off). A legend on the right indicates that a solid dot represents 'on' and a crossed dot represents 'off'. Red circles highlight the nodes that are added or modified during the process. An arrow points from the left towards the grid.

VLSI Design, Verification and Test 38

So for each cube if add those sub cubes that in if that are not covered by other sub cubes that are not covered by others of you so we are saying that this cube this vertex is covered by cube this and cube this.

(Refer Slide Time: 19:34)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

01 11
00 10

01 11
00 10

● on
⊗ off

VLSI Design: Verification and Test 38

However this vertex or this implicant.

(Refer Slide Time: 19:38)

Espresso Algorithm – Reduce Gasp

- **reduce_gasp (F, D)**
- For each cube in F add those sub-cubes of cubes in F that are not covered by other cubes.
- It uses D to ensure that these new cubes are not produced for just some DC nodes.

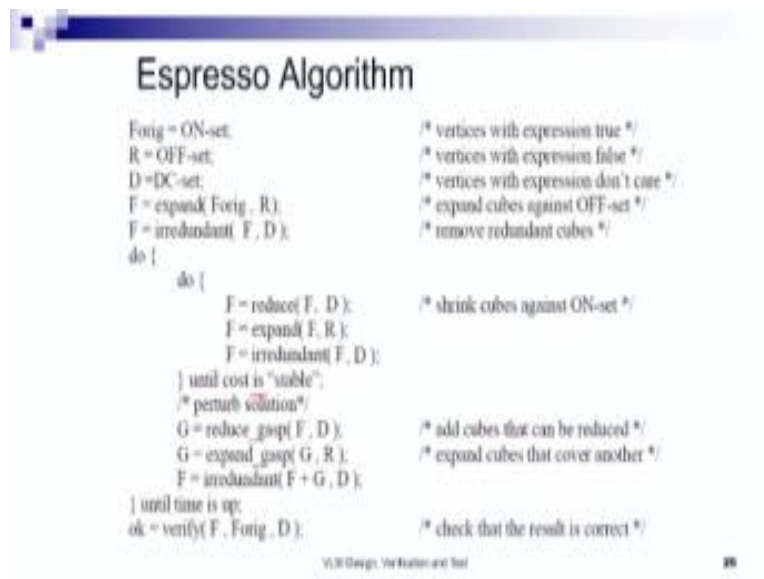
The diagram shows a 2x2 grid of nodes. The top row nodes are labeled '01' and '11', and the bottom row nodes are labeled '00' and '10'. A legend indicates that a filled circle represents 'on' and a crossed circle represents 'off'. In the initial state, the '00' node is 'off' and the '10' node is 'off'. Red circles and arrows highlight the process of adding sub-cubes not covered by others. In the final state, the '00' node is 'on' and the '10' node is 'off'. The '01' and '11' nodes are 'on'.

VLSI Design, Verification and Test 38

Is not covered by any other covered in f any other implicant in f and therefore we add this to f this is what Jack does it uses d to ensure that these new cubes are not produced for just some do not Cairn owns the new cube that we form that we add should not consist of only don't care nodes then we come to the last step the expand cast this function takes as input the set G produced after reduced gasp and on.

So this function takes as input what reduced gasp has produced me so now in F we have many others hub cubes right we have many other sub cubes which have been obtained after this reduction and addition of cubes that are not covered by other cubes right now we expand from all the cubes and adds them if they covered other cubes so r is used to ensure that nodes in the oxide are not included right the result the redundant cubes introduced are removed later so these are the basic functions of the espresso algorithm and we will now take a look at the detailed algorithm.

(Refer Slide Time: 21:02)

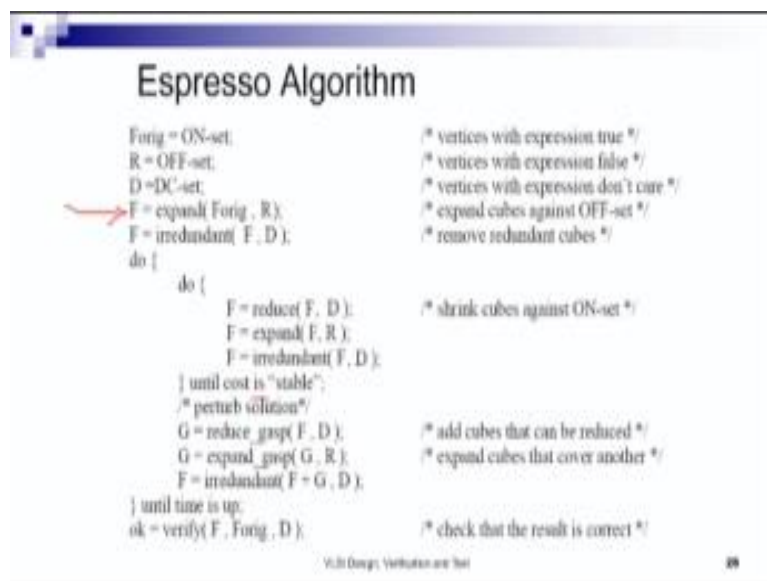
A slide titled "Espresso Algorithm" with a blue header bar. The slide contains a list of algorithm steps with comments. The steps are: 1. Forig = ON-set; /* vertices with expression true */ 2. R = OFF-set; /* vertices with expression false */ 3. D = DC-set; /* vertices with expression don't care */ 4. F = expand(Forig, R); /* expand cubes against OFF-set */ 5. F = irredundant(F, D); /* remove redundant cubes */ 6. do { 7. do { 8. F = reduce(F, D); /* shrink cubes against ON-set */ 9. F = expand(F, R); 10. F = irredundant(F, D); 11. } until cost is "stable"; /* perturb solution */ 12. G = reduce_gasp(F, D); /* add cubes that can be reduced */ 13. G = expand_gasp(G, R); /* expand cubes that cover another */ 14. F = irredundant(F + G, D); 15. } until time is up; 16. ok = verify(F, Forig, D); /* check that the result is correct */ At the bottom of the slide, there is a small text "VLSI Design: Verilog and Test" and a page number "28".

```
Espresso Algorithm

Forig = ON-set; /* vertices with expression true */
R = OFF-set; /* vertices with expression false */
D = DC-set; /* vertices with expression don't care */
F = expand(Forig, R); /* expand cubes against OFF-set */
F = irredundant(F, D); /* remove redundant cubes */
do {
  do {
    F = reduce(F, D); /* shrink cubes against ON-set */
    F = expand(F, R);
    F = irredundant(F, D);
  } until cost is "stable"; /* perturb solution */
  G = reduce_gasp(F, D); /* add cubes that can be reduced */
  G = expand_gasp(G, R); /* expand cubes that cover another */
  F = irredundant(F + G, D);
} until time is up;
ok = verify(F, Forig, D); /* check that the result is correct */
```

Once more as we said the algorithm takes a given expression in binary variables as input and represents it in the form of a hypercube the onset represents the nodes representing min terms which are one when the expression is one the offset represents the node the offset represents our nodes representing min terms which are zero when the expression is one the DC set consists of nodes representing the min terms which are don't care values initially the cubes in the onset consist of individual nodes in the onset now the algorithm begins by expanding the onset against the offset so this is where the algorithm begins.

(Refer Slide Time: 21:52)

The slide displays the Espresso Algorithm in a code-like format. It starts with defining Forig as the ON-set, R as the OFF-set, and D as the DC-set. A red arrow points to the line 'F = expand(Forig, R);'. The algorithm then enters a 'do' loop. Inside this loop, there is an inner 'do' loop containing 'F = reduce(F, D);', 'F = expand(F, R);', and 'F = irredundant(F, D);'. The inner loop continues until the cost is 'stable'. After the inner loop, there is a comment '/* perturb solution */', followed by 'G = reduce_gasp(F, D);', 'G = expand_gasp(G, R);', and 'F = irredundant(F + G, D);'. The outer loop ends with '/* until time is up.' and 'ok = verify(F, Forig, D);'. Comments on the right side explain each step: '/* vertices with expression true */', '/* vertices with expression false */', '/* vertices with expression don't care */', '/* expand cubes against OFF-set */', '/* remove redundant cubes */', '/* shrink cubes against ON-set */', '/* add cubes that can be reduced */', '/* expand cubes that cover another */', and '/* check that the result is correct */'. The slide footer includes 'VLSI Design, Verification and Test' and the number '28'.

So this is where the algorithm begins it begins by expanding the onset against the offset and the cubes are expanded by adding neighbouring onset nodes or DC nodes this what expands the tubes are expanded by adding neighbouring onset nodes or DC nodes once a direction of expansion is chosen once the direction of expansion is chosen then it is fixed and the cube can expand further only in that direction this is what expand does so it keeps the direction of expansion fixed when you cannot expand in that direction anymore to obtain a prime you say that it is to expand in the direction anymore.

If you obtain a prime implicant right once all cubes have been maximally expanded the redundant method is applied so once this expansion step is done once all cube has been maximally expanded we apply the redundant step okay it chooses the essential cubes only those covering nodes not covered by other nodes it then removes those cubes all of whose ones are covered by some other prime implicant it removes all those loans.

This is what the Erie London step does of the remaining cubes those are chosen which result in a minimal cover next the reduced expression is applied it shrinks larger cubes so after this eerie dundon. We go inside this loop we go inside the slope we apply the reduce so what does reduced to it shrinks larger cubes into smaller ones wherever possible the resulting news hub cubes can then be expanded in a direction different from which they were expanded previously and this probably will give me a better solution.

This lets different min terms to be combined to produce possibly larger clips so when we expand in a different direction we will possibly get different prime implicants covering a different set of minterms and these this prime could be larger than the previous prime implicant that we get. So why do we need to do all these steps because finally we want to get a set of prime implicants which covers all the onset of the function and this cover is minimal In the set it cannot in the sense it contains the minimum number of prime implicants in it.

This is what is done if a cube cannot be expanded in your new direction then it is re expansion in the previous direction the ear attendant tubes are then removed this cycle of reduced expand remove redundant is repeated till the cost is stable that is the expression cannot be further reduced next a simulated annealing type approaches we discussed is applied the intermediate solution is quartered by adding new cubes these are sub Q of existing Q's right these are sub Q of existing cubes which are not covered by other cubes.

This is what the deuce gas does the new set is then expanded then the redundant cubes of the union of the new set and the immediate intermediate solution are removed this is what happens in this step this new set is subjected to the same reduce expand redundant iterations this goes on the above procedure is repeated for some predetermined time right.

So this whole thing is ready total Imperium repeated for some predetermined time until time is up to get a final solution so espresso is an anytime solution is called an anytime solution because the algorithm that we get can stop can be stopped at any time and we will get a solution so the more time we give to the algorithm the solution is expected to be better and better but at any point in time if we stop we will get a solution hence it's an anytime solution okay.

So a new solution can be obtained by perturbation of an old solution we can repeat the above process while maintaining the best solution we obtain so at any point in time we always keep the best solution so that at the end of the whole process after all the iterations that we have done we ultimately get the final best solution the final discovery and the minimum logic expression.

So this is was an overview of a heuristic methodology for 2 level logic minimization this is a standard tool that is used in current the VLSI design and with this we we have taken a brief look at the logic synthesis step in the VLSI design process we come to the end of this module with this discussion.

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