Synthesis of Digital Systems

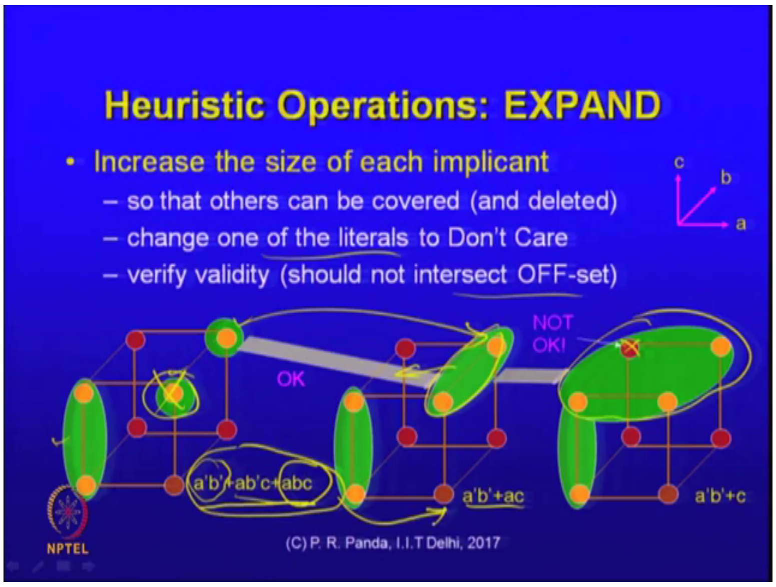
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 22
Two–level Logic Optimisation

So, this two level logic minimization, it requires some heuristic sort of approach just because enumeration of the prime implicants will take too much time there are too many of them. The general approach is actually quite simple there will be some kind of loop we will look at what those operations are, but there are some simple operations which hopefully might take us in the direction of an optimized function starting from some input function.

So, we start with a cover and just attempt to improve it in different ways what we will be studying as part of this strategy is; what are those improvement operations?

(Refer Slide Time: 01:15)



Heuristic Operations: EXPAND

- Increase the size of each implicant
  - so that others can be covered (and deleted)
  - change one of the literals to Don't Care
  - verify validity (should not intersect OFF-set)

(C) P. R. Panda, I.I.T Delhi, 2017

So, let us take a little more detail look at some of those operations first of which is an expansion this just refers to the implicants that are given to us that we start off with, and they covered that n dimensional space where n is the number of Boolean variables in terms of which the function is composed.

And these implicants occupy some sizes depending on the number of literals that is there in each of the implicants. So, if you have an implicant like that; then it occupies one vertex this one occupies corresponding to these two a d that implicant occupies 2 vertices each of them corresponds to some region in that cube.

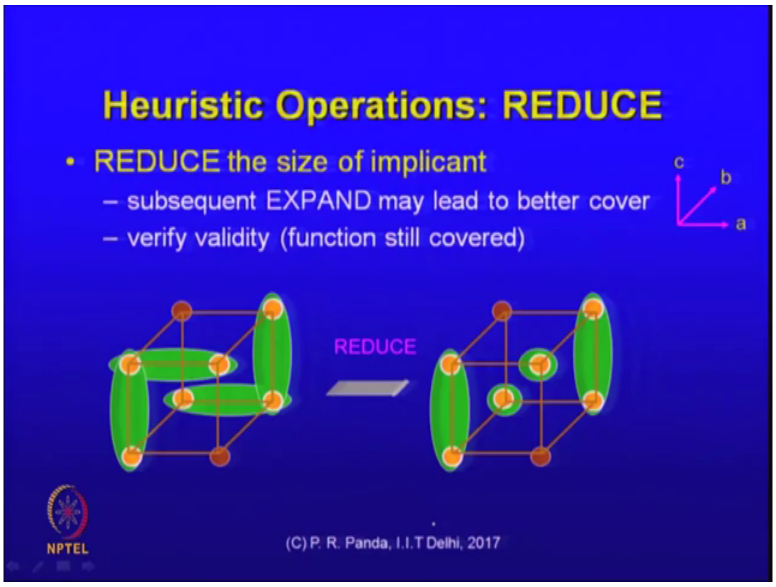
In the point of the expansion, operation is to just try and increase the size of each implicant ok. Why do that that reduces the number of literals of course, when you expand the size, but remember the objective of the two level minimization is not merely to reduce the number of literals, it is to reduce the number of cubes number of product terms, but the point is when you expand one of those literals in that direction where it occupies now a larger space.

It is possible that in the process you might cover some of the already existing implicants if that happens then you could delete the one that you have just covered as a result of the expansion. So, in this example as we increase the size of this it turned out that the new one actually now covers the existing implicant. So, we can drop that. So, the number of implicants has reduced from 3 to 2 that buys us something in this two level that is the process and in terms of the implementation of this. So, diagrammatically this should be clear how to implement this is also an interesting problem we will get to later on, but logically what we are doing is we are changing one of the literals in that implicant to do not care we are dropping one of those literals.

In the process of course, one should also verify that this is a valid expansion, valid expansion means it is if it covers something in the onset; if it covers something in they do not care set, but it should not cover something in the offset if it does then that would be an invalid expansion, but that is all that is there as the basis of this expand operation you pick up some variable if this term has three variables then along a along b each represents a direction a, b and c all three represent some direction along which you can perform the expansion.

We can try out that expansion in different directions, and as long as it does not intersect any vertex in the offset, it is a valid expansion we can continue the expansion. So, a second time if you try to now expand that cube in this direction; thereby leading to this region there this is an invalid expansion because in the process you would be covering a vertex that is there in the offset. So, what this is telling us is you start with a function like that this is an optimized version of that function as a result of this expand just this one operation.

(Refer Slide Time: 05:32)



Heuristic Operations: REDUCE

- REDUCE the size of implicant
  - subsequent EXPAND may lead to better cover
  - verify validity (function still covered)

(C) P. R. Panda, I.I.T Delhi, 2017

We could also go the other way around reduce is the reverse you are reducing the size of the implicant reduce here refers to the region that the implicant covers in that space.

The way this comes about is you add a variable to that cube of course, the smaller the size of the cube in terms of the number of variables that are there in it number of literals that are there in it the larger the space that it occupies in that space. So, this reduce just refers to in the dimensional space how much space it covers? How many vertices it covers? What would be the point of the radius the point of expansion is that it improves it ultimately reduces the number of product terms we have which is a good thing.

This operation does not lead to a more optimized solution right actually by itself neither expand nor reduce actually lead to any difference in the number of cubes that are there this we are just changing the size of the cubes, but in the process of expansion later on you can drop some cube that is there and that would lead to a reduction in the number of cubes, here too we are changing the size of an implicant that does not increase or decrease the number of implicants we have.

Which is the ultimate objective of this optimization reducing the number of implicates, but this itself does not necessarily lead to any improvement as it is, but it could be that you combine it with a subsequent expand operation that might lead to a better cover, it is just an elementary operation that by itself does not lead to a more optimized solution. In fact, it most probably is leading to a less optimized solution because it is reducing the size of course; our objective in all of this logic minimization is that we should have these individual cubes to be as large as possible not as small as possible. So, this could only be thought of as an intermediate step that may help us in a different expansion that might be better.

Student: sir in are you are you implying that in expansion step wherever we expand a particular implicant, it should not overlap with an existing larger implicant is there a restriction of such.

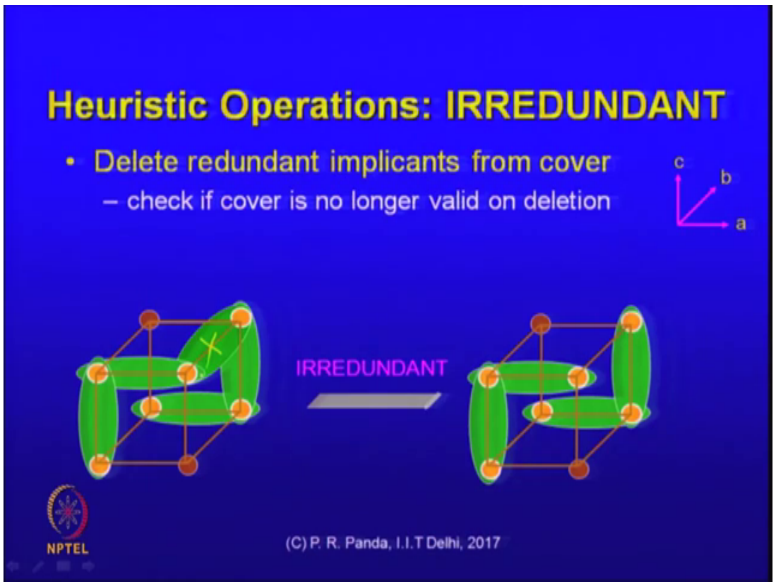
No, no, I actually only covered the definition so far, how it will be part of an algorithm we will look at, but I already hinted that that overall algorithm may not be very complex; it is like a loop where you run these elementary operations until there is no further improvement.

Student: It is hard to imagine why a reduced can help us do a better expansion.

We will get to that ah, but so far let us just define the individual operations knowing fully well that the last step in the algorithm cannot be reduced.

Student: Cannot be reduced.

(Refer Slide Time: 08:54)



Heuristic Operations: IRREDUNDANT

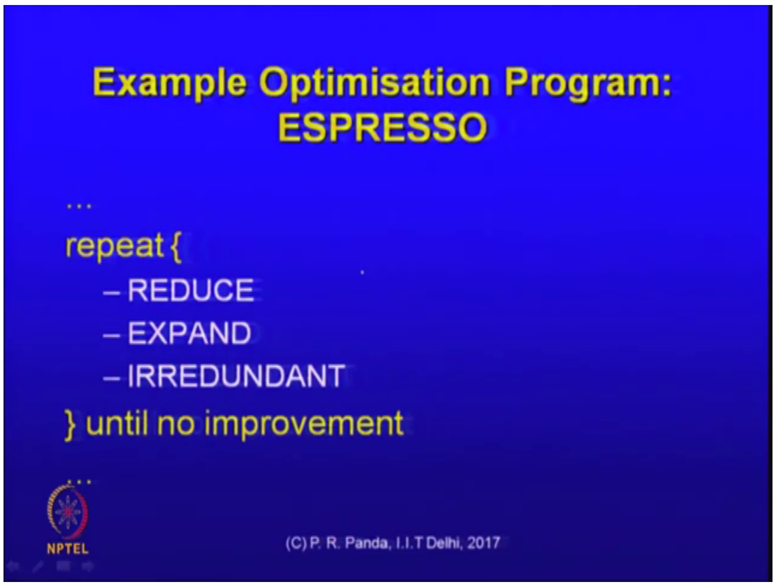
- Delete redundant implicants from cover
  - check if cover is no longer valid on deletion

(C) P. R. Panda, I.I.T Delhi, 2017

Yeah, it does not help; obviously, the third one is called irredundant that deletes some implicants from the cover that are established to be redundant, you could think of this as following the expand step in the process of expansion you might have rendered some implicants redundant, because they got covered by the expansion process then we can delete that the process of identifying that and removing that is what is included in the irredundant step.

What do you do as a result of this, you need to check that when you drop one of those implicants the cover is still valid which is in our example here, we chose to drop this implicant ok. In the remaining implicants after the dropping of that particular implicant, it must be that all the points in the onset are still covered, if they are not; then it is a wrong function so that the check needs to be done in the process of identifying an implicant to be dropped.

(Refer Slide Time: 10:18)



Example Optimisation Program: ESPRESSO

```
...
repeat{
    – REDUCE
    – EXPAND
    – IRREDUNDANT
} until no improvement
...
```

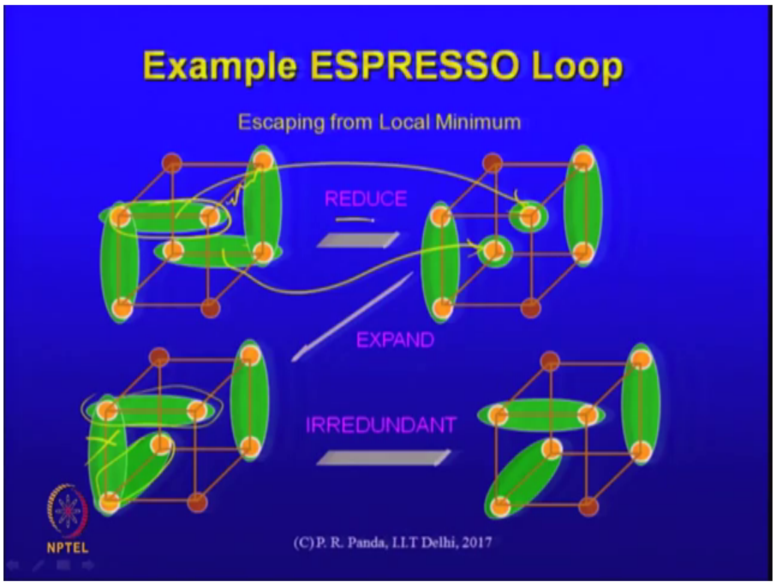
(C) P. R. Panda, I.I.T Delhi, 2017

Yes, but an overall algorithm is not too complex you can think of a lot of variations the espresso is an example of an early logic optimization tool, where you are essentially doing the following reduce expand and irredundant in a loop until there is no further improvement. Why do you expect that loop to help you perform reduction you perform expansion and in the process maybe an implicant got dropped and we dropped some implement that irredundant step might result in what we called an irredundant cover.

That is you cannot drop any more implicants from there and still manage to cover the function, why do we expect a second iteration to result in a different solution, we will maybe understand in a little more detail as we look at how expansion is done what is the issue involved there and, and reduce and so on, but it turns out that you have a lot of choices when you do expand which of the available implicants, there is a set of implicants that you are trying to drop some of them now what is the order in which you should process those implicates, that is one example it is not clear.

The quality of the solution depends on the order in which you do that processing, the other thing is even after you choose the implicant to process what direction should you expand in, because there are n variables you could possibly expand it in n different directions. That too we need to discuss a little bit, but what might happen is that in a future iteration you can try expansion along a different direction, that is why iterations are there in this process the number of choices is large for each of these steps for reduce for expand and for irredundant also.

(Refer Slide Time: 12:43)


Example ESPRESSO Loop
Escaping from Local Minimum
REDUCE
EXPAND
IRREDUNDANT
NPTEL
(C) P. R. Panda, I.I.T Delhi, 2017

But in the process we might actually end up hopefully escaping from some local minimum remember that irredundant was leading us to a local minimum not guaranteed to be optimal, but this is an example of a possibly transformed version of what originally was given originally we started of with 5 implicants, but we dropped one of them and this is any redundant cover not necessarily optimal we have already seen this; yeah.

Student: I was thinking that, so in the last when. So, we are number of directions to be (Refer Time: 13:20), right, there will be lot of iterations which we can do.

Yeah.

Student: So, if I plot the graph of improvements over each iteration it need not to be an increasing or decreasing right can be a.

Yeah, yeah, it could be.

Student: So, something like that, so it is more of like we need to look for global maxima.

I need to keep track of the best solutions, so far in each iteration we can explore different directions, but in the process we can keep track of what is the best result that we have got.

Student: So, we can keep and we might have to come back then so, because if I just.

Yeah, we would have to come back, yeah, we would ultimately of course, come back and choose the 1 that gave us the best results, but this is a branching structure of the solution you may need to come back to a previous, iteration that looked more promising and explores that one further.

Student: So, there I map this problem to so if we look at the extended algorithms of our example graph partitioning problem so there.

Yeah.

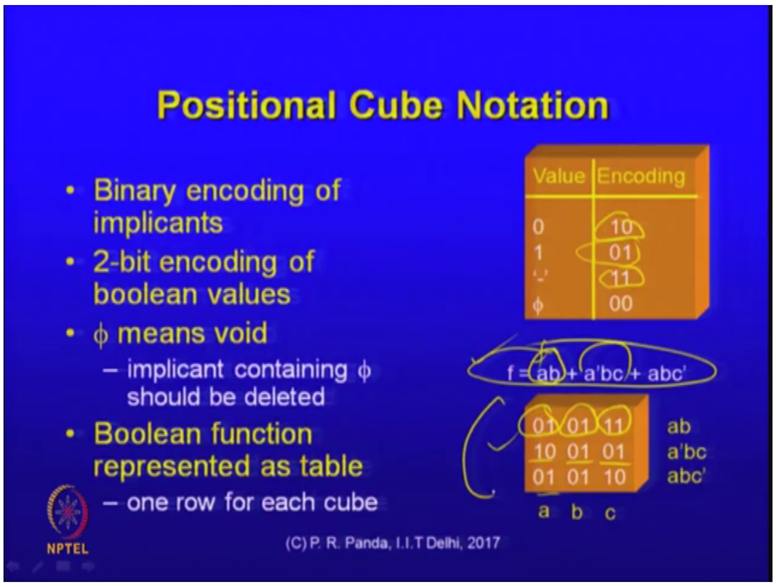
Student: The problem is more of finding a, so you keep on iterating and you find the maxima you come back now this was better.

Yeah, yeah each of the levels may correspond to further improvement after having chosen an intermediate solution that looks promising initial iterations you choose some directions all the directions are not necessarily promising, but you could evaluate the result of individual expansion along different directions and maybe choose for further exploration those directions that seem more promising yeah and the standard search strategy is branch and bound and so on, that are there; obviously, apply to algorithms like this yes by itself, this loop is really not doing anything; it is just invoking those operations again and again the intelligence lays in what direction you choose for each of those individual operations ok.

So, it is possible that you have this kind of a solution that is kind of a local minimum, because you cannot drop any implicants further from this, but if you apply reduce to this maybe that implicant there is reduced to this implicant here is reduced to this could lead to an expansion possibly in a different direction from what was tried earlier, initially we had that being covered, now we can have possibly this is an expansion this is an expansion.

So, I need to somehow keep track of what expansions I have done in the past. So, that in future iterations I can traverse a different direction in the search space, but if I do this expansion then you can see that there is one implicant there that could be dropped on the application of the irredundant operation, I have a result that is better than what I started off with, that is the way in which reduce might make difference again by itself that first step did not really change anything with respect to quality of the solution, but it enabled expand to occur in a different direction that is what a typical loop of this algorithm, right.

(Refer Slide Time: 17:20)



Positional Cube Notation

- Binary encoding of implicants
- 2-bit encoding of boolean values
- ϕ means void
  - implicant containing ϕ should be deleted
- Boolean function represented as table
  - one row for each cube

Value	Encoding
0	10
1	01
'-'	11
ϕ	00

$f = ab + a'bc + abc'$

a	b	c	
01	01	11	ab
10	01	01	a'bc
01	01	10	abc'

(C) P. R. Panda, I.I.T Delhi, 2017

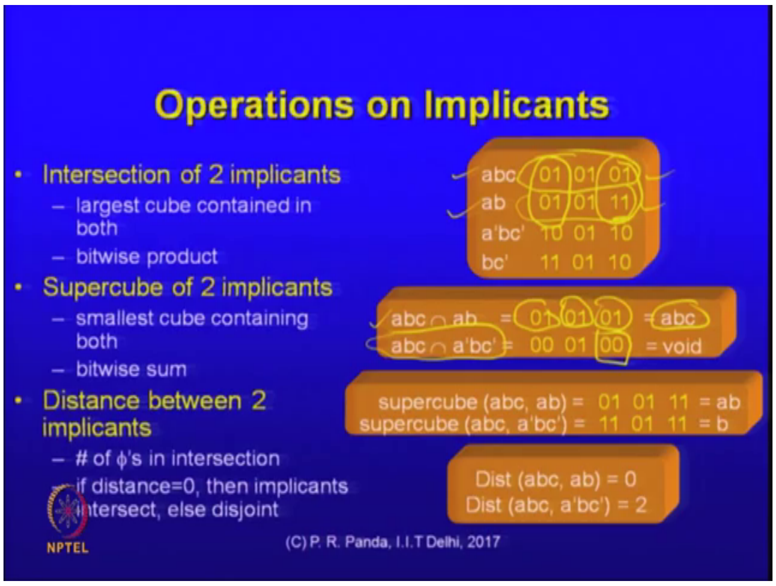
This was a visual illustration, it is representation and implementation is also interesting you have to be a little careful in implementing those operations because they are invoked. So, many times, it is important that the implementation of those steps should be efficient while this is not the only way to do it this illustrates some of the issues that are involved.

So, let us introduce this positional cube notation I have a function like this, how do I represent it we have talked about this off and on we did not really get to the actual representation. This positional cube notation could use a binary encoding of a particular manner for the implicants, their encoding could look like this if my variable is there in uncomplimentary form normal form like a then I use as 0 1 2 encode it why 2 bits it is because I also want to take care of the do not care condition.

So, if it was just 0 and 1 then 1 bit would have been enough, but I have a third thing to take care of and which also leads to a fourth encoding that we have some interesting use for, but the presence of an a is captured by a 0 1 presence of an a prime would be captured by a 1 0. If both of these are 1, then it means it said do not care and if both of them are 0 then it means it is a void or a null that has some interesting uses that we will see soon, but a function like there is a Boolean function like this could be captured in this way, where do you have 1 row for each cube.

So, the three cubes here in the function then you have three rows and you have 2 bits for each variable. So, if a b would be represented as 0 1 for a 0 1 for b and 1 1 for c meaning that c does not appear there it is it is do not care with respect to c, that particular cube is a prime b c would be 1 0 because a is complemented 0 1 because b is normal and 0 1 because c is normal 1 complimented. So, that is what the representation would be for the Boolean function that we start off with.

(Refer Slide Time: 20:19)



Operations on Implicants

- Intersection of 2 implicants
  - largest cube contained in both
  - bitwise product
- Supercube of 2 implicants
  - smallest cube containing both
  - bitwise sum
- Distance between 2 implicants
  - # of ϕ's in intersection
  - if distance=0, then implicants intersect, else disjoint

abc	01	01	01
ab	01	01	11
a'bc'	10	01	10
bc'	11	01	10

abc ∩ ab = 01 01 01 = abc

abc ∩ a'bc' = 00 01 00 = void

supercube (abc, ab) = 01 01 11 = ab

supercube (abc, a'bc') = 11 01 11 = b

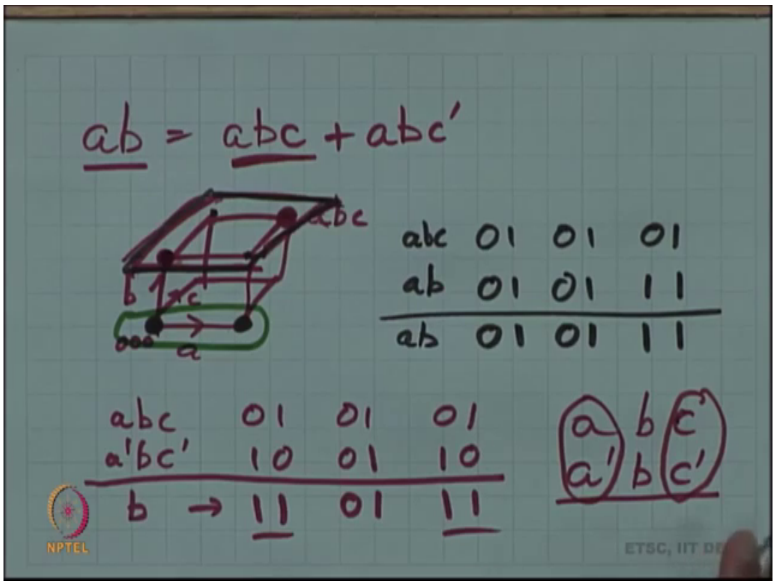
Dist (abc, ab) = 0

Dist (abc, a'bc') = 2

(C) P. R. Panda, I.I.T Delhi, 2017

We can think of a few operations with that particular representation, let us define an intersection of two implicants intersection would be defined as the largest cube that is contained in both of them right an example would be. Let us say I have a cube a b c that is represented like this 0 1 0 all 0 1 a b is represented like this, there is 1 1 here at the c position because he is not there, if I take the intersection of a b c and a b I should get what a b remember is nothing, but the sum of these 2 in terms, right those 2 i f c were the third variable that I have that would be the 2.

(Refer Slide Time: 21:12)



$\underline{ab} = \underline{abc} + abc'$

abc	01	01	01
ab	01	01	11
ab	01	01	11

abc	01	01	01
a'bc'	10	01	10
b →	11	01	11

a b c

a' b c'

So, a b c if you intersect, so anyway it is a function of three variables. So, a b corresponds to 2 vertices in that Boolean space a b c corresponds to 1 vertex the intersection of these two would be what.

Student: 1 vertex.

Would be this vertex.

Student: Yeah.

Right; it would be that vertex, so that is what the intersection essentially is the set of vertices that are common between the 2 cubes right. This would be captured in our representation as follows you look at the encoding for a b c take the encoding for a b and you just do a bitwise, product of all the 6 columns. What do you get 0 1 is intersected with the 0 1 you get a 0 1 all the corresponding places are intersected right, you take product of the here also you get 0 1, but the intersection of these two gives us 0 1, it is the bitwise product.

So, I essentially have recovered a b c which is the right intersection of those 2 implicants about this intersection. If I take a b c and intersect with a prime b c prime, what do I expect remember that space is like this is 3 dimensional and this is a direction that is b direction and this is c direction let us say this is 0 0 0, then a b c corresponds to that point a prime b c prime corresponds to may be this point maybe at this point what is the intersection of those two regions.

Student: Null void.

There is no intersection these regions are not overlapping. So, there is no intersection that is captured in our operation here by the presence of a 0 0 in the intersection when you do that bitwise product, you take the product of 0 1 with 1 0 you get 0 0. And the existence of a single null literal anywhere there just tells us that this is what there is nothing in that set it represents a set of points and 0 0 tells us that it is void.

Which is again a correct capture of what we expect in that intersection, correspondingly we can define a super cube of 2 implicants that is the other way around it is the smallest cube containing both the implicants. So, union, but it is not merely the union because you take again in this example of this being 1 implicant and that being the other implicant what is the smallest cube that contains both the implements.

Student: Both of them.

There has to be a cube still. In fact, this is the smallest cube that includes both of those implicants. Yeah, I need a single implicant that is why we do not call it a union here we are calling it a super cube because it does include some other minterms that were not there in the original 2 cubes, that is why different term is chosen as opposed to union which of course, has a connotation of the set of the minterms that are there in both of them.

That is a different operation, but that is not the same as one region that union of 2 cubes might result in multiple cubes you might not get a clean single cube that defines the union sometimes you might, but sometimes you might not if it were these two points, if you started off at that point and that point then the union could be the same as the super cube because you would get essentially a single cube that represents the union.

But it is computation should be easy we just take the bitwise or of the 2 implicants and I would get my super cube. So, between a b c and a b if you take the bitwise or you essentially get zeroes here 1 0 1 the third literal there would result in 1 1, because that is the union this is 0 0 1 0 1 0 1 for a b c and for a b I have 0 1. The super cube would be the bitwise sum of these which is 0 here 1 here 1 here 1 1; what cube is this?

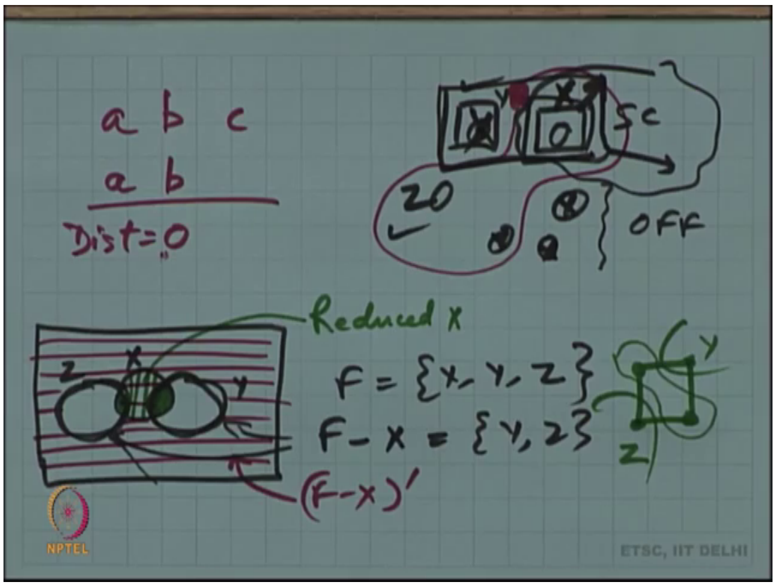
Student: a b.

This is a b itself, this is fine right if there is a containment relationship between the 2 cubes one is a subset of the other the super cube is just a larger cube that is what you would get if you just do it a bitwise or operation, but like we saw if you take a b c and a prime b c prime, which could be these 2 nodes then you have a super cube where you had a b c was 0 1 0 0 1. So, a prime b c prime was 1 0 0 1 for b and 1 0 for c prime.

The super cube would be the following it would have 1 1 in both of these places right this place has a 0 1. So, the corresponding implicant here is just b because this I would leave out and this I would leave out and that really is this example that we have here this is an example, where the super cube does have some minterms that were not there in the original 2 cubes.

So that is intersection and super cube distance between 2 implicants is just defined as the number of positions in which they are different. So, the difference being computed like this you have a b c, and let us say a prime b c prime then will require the distance to be 2 because these 2 literals are different on the other hand.

(Refer Slide Time: 29:33)



If you had a b c and you have a b then this distance is 0 this distance is 0. So, all three this is a do not care and that do not care anyway there is an inclusive relationship, so we will say that the distance here is.

Student: 0.

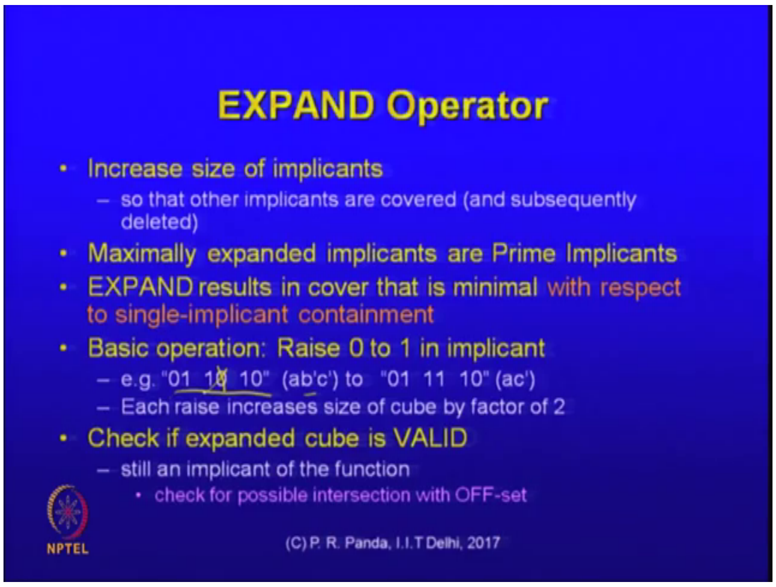
0; this is captured by performing an intersection of the 2 implicants and counting the number of.

Student: Voids.

Voids right, so those are the operations you can now see the reason why this notation was chosen in a way as to enable these operations for example, because the intersection in super cube these operations are going to be performed again and again we want them to be efficient.

And if I can implement those operations as bitwise products or sums you know that the actual translation in 2 instructions and so on might be very efficient and therefore, that is the basis of the choice of this encoding of implicants. Ultimately it might translate to just a single instruction logical and or a logical or if I manage to for example, capture the implicant within one word if I managed to capture it within 32 bits of a processor then this can actually translate nicely into just a single instruction.

(Refer Slide Time: 31:17)



EXPAND Operator

- Increase size of implicants
  - so that other implicants are covered (and subsequently deleted)
- Maximally expanded implicants are Prime Implicants
- EXPAND results in cover that is minimal with respect to single-implicant containment
- Basic operation: Raise 0 to 1 in implicant
  - e.g. "01 10 10" (ab'c') to "01 11 10" (ac')
  - Each raise increases size of cube by factor of 2
- Check if expanded cube is VALID
  - still an implicant of the function
    - check for possible intersection with OFF-set

(C) P. R. Panda, I.I.T Delhi, 2017

So with that as the representation, how do I implement the expand operator the idea is to increase the size of the implicants. So, that later on other implicants are covered the covering and actually deleting of the other implements that got covered is not part of the expand process it is part of the irredundant process, that also is not trivial because you have to choose which subset of things that could be dropped would you actually drop because you cannot drop all of them, right.

So, that is also a separate two interesting problem, but expand itself is not necessarily trivial idea is trivial, but what are the problem is that choices are there what are the choices, these maximally expanded implicants are the prime implicants, you cannot expand anymore without intersecting the offset means that you have arrived at a prime implicant.

So, the expand process would result in a cover that is minimal with respect to single implicant containment, single implicant containment means. In the process of that expansion, you can make sure that that particular implicant that you are processing will not be covered at the end of that process if you keep expanding, at the end you will come up with an implicant that it cannot; obviously, be contained in any other implicit that is the nature of the transformation of course, when do you stop you stop when you are intersecting the offset which means that further expansion is not really possible right.

So, our basic operation then would be in terms of that encoding you raise one of the zeros to a 1 in the implicant, we want to drop one of the literals right either a or a prime if it exists we want to drop. So, here this is what let what implicant is this is a b prime c prime, expanding it means that I convert this let us say to a c prime means I am dropping the b prime literal, conversely I may drop a and that might become b prime c prime, either way this basic operation is one of those zeros is converted into a 1 right, dropping it means that a 1 0 or a 0 1 becomes 1.

So, a 0 is converted into a 1 as the basic operation each such raise increases the size of the cube by a factor of 2, 1 lateral is dropped means the size in that space has doubled right. I need to check if that expanded cube still valid which means that I checked for a possible intersection with the offset.

Student: Yes.

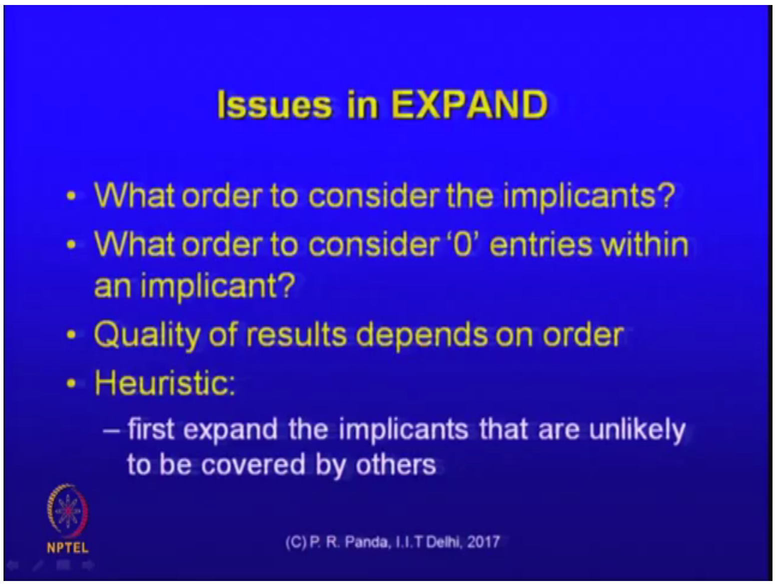
Assuming that this offset is available to me in the form of some set of implicants, I can quickly do an intersection of the expanded cube with all the cubes that are given and together they contain the offset or if the offset is given as a set of just all the min minterms that is fine with each of them I can quickly do an intersection well quickly is a little tricky, because the operation is quick with 1 min term or with 1 cube the number of such cubes might still be.

Student: large.

Large; but the validity check is essentially an intersection operation with all of the;

Student: Offsets.

(Refer Slide Time: 35:10)



Issues in EXPAND

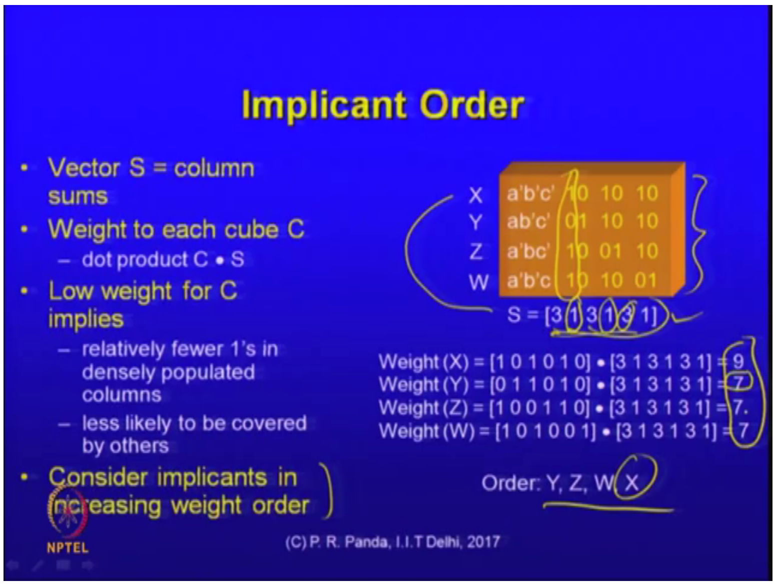
- What order to consider the implicants?
- What order to consider '0' entries within an implicant?
- Quality of results depends on order
- Heuristic:
  - first expand the implicants that are unlikely to be covered by others

(C) P. R. Panda, I.I.T Delhi, 2017

Min minterms in the offset, so that is the operation in terms of the implementation what are the issues here. Issues are what order to consider the implicants I have a large number of implicants and the idea of course, is to take an implicant and increase it is size until it does not intersect anything in the offset that is fine that is easy to do, but the thing is when I have a large number of implicants, then which one do I start with that does influence the later steps and therefore, the quality of my solution.

Other thing is having chosen that single implicant what order do I raise those 0 entries to 1, that 2 is not obvious there you have a large number of variables. So, all of them are candidates and which ones to drop which direction that expands should in that itself is not obvious both of them are questions that are open, and we somehow have to use some heuristic to choose promising implicants and promising directions within those implicants. The heuristic could be that we expand those implicants first that are unlikely to be covered by others let us just quantify that with an example.

(Refer Slide Time: 36:43)



Implicant Order

- Vector S = column sums
- Weight to each cube C
  - dot product C • S
- Low weight for C implies
  - relatively fewer 1's in densely populated columns
  - less likely to be covered by others
- Consider implicants in increasing weight order

X	a'b'c'	10 10 10
Y	ab'c'	01 10 10
Z	a'bc'	10 01 10
W	a'b'c	10 10 01

S = [3 1 3 1 3 1]

Weight (X) = [1 0 1 0 1 0] • [3 1 3 1 3 1] = 9
Weight (Y) = [0 1 1 0 1 0] • [3 1 3 1 3 1] = 7
Weight (Z) = [1 0 0 1 1 0] • [3 1 3 1 3 1] = 7.
Weight (W) = [1 0 1 0 0 1] • [3 1 3 1 3 1] = 7

Order: Y, Z, W, X

NPTEL

(C) P. R. Panda, I.I.T Delhi, 2017

Here is my set of implicants x y z and w and that is the table for those implicants the question is which one is unlikely to. So, which one should I start processing first x y z w. Here is one simple argument you align them in that table and form a vector of the sum of each of those columns and there are 3 1 there, so that number is 3.

There is a single one here that number is 1 the 3 ones here, which results in a 3. So, that is the sum now you take the dot product of that sum with each of the individual implicants each cube you get something. So, that is what we are saying is a weight if you take the dot product of x with us you get some number here 9 x is this. So, where did the 9 come from essentially 3 3 and 3, where it is 1 these numbers are picked up yeah where this is 1 the corresponding number is picked up from the sum. So, you get 9 for x, so the implicant itself you take the dot product with the some vector here and you would get some weight here that is what we are calling the weight.

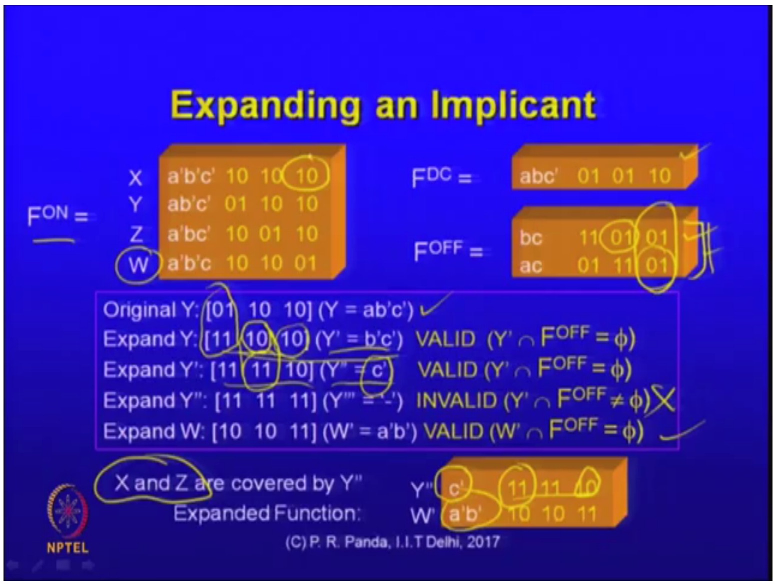
The operation clear the dot product what is the meaning of this if the weight is low for one of those cubes, then it means that in the densely populated columns that cube actually does not have a 1. Otherwise it would have picked up the 3 densely populated means consider a column and there are large number of ones out there if that cube also had a 1 then the larger number would have been picked up in the weight if it is relatively small it means that it has a 0 in the places that are densely populated those columns that are densely populated.

The argument is that if that number is low if that weight is low then because too many other cubes do not exist that have a one in that space in that particular column, that cube is less likely to be covered by others that is a heuristics argument if that number is high it means that many others have a one there. So, it is more likely that it will be covered, why is it one in that column it is because our literal if it is a then either that cube has an a or it is actually do not care. right.

Both of which are favourable with respect to covering our implement right if it is conflicting if it is a 0 then it will not cover. So, the weight will be low if there are more zeros in that column relatively speaking that is the intuition. So, less likely to be covered means that we can consider the implicants in increasing weight order. So, our first cubes to be processed could be the ones that have low wait because those cubes are likely to stay in the final result also right, but the other ones because there is a relatively larger overlap those are candidates for dropping.

So, since they are candidates for dropping let us try to drop them and instead let us try to expand the cubes that are less likely to be covered by others. So, that is why that order is what is resulting point is that we want to consider x last, in this particular examples x y y is that and w it does not matter the weight it is the same, but x comes later.

(Refer Slide Time: 41:16)



Expanding an Implicant

$F^{ON} =$

X	a'b'c'	10	10	10
Y	ab'c'	01	10	10
Z	a'bc'	10	01	10
W	a'b'c	10	10	01

$F^{DC} =$ abc' 01 01 10

$F^{OFF} =$

bc	11	01	01
ac	01	11	01

Original Y: [01 10 10] (Y = ab'c') ✓
Expand Y: [11 10 10] (Y' = b'c') VALID (Y' $\cap$ $F^{OFF} = \phi$)
Expand Y': [11 11 10] (Y'' = c') VALID (Y' $\cap$ $F^{OFF} = \phi$)
Expand Y'': [11 11 11] (Y''' = '-') INVALID (Y' $\cap$ $F^{OFF} \neq \phi$) X
Expand W: [10 10 11] (W' = a'b') VALID (W' $\cap$ $F^{OFF} = \phi$) ✓

X and Z are covered by Y''

Expanded Function:

Y''	c'	11	11	10
W'	a'b'	10	10	11

(C) P. R. Panda, I.I.T Delhi, 2017

So, if I go ahead and expand an implicant; let us see what it looks like. So, originally the onset was these; there is I do not care set and there is an offset yeah one more point, that

I did not specify it belongs somewhere there should be 8 vertices or 8 minterms of course, actually the b c is fine these are actually cubes in the offset.

So, that is fine all the 8 might actually recovered here, since I picked up the y say for expansion let me expand y along the a direction that would lead to the 0 1 becoming a 1 1. Correspondingly that would lead to the new expanded implicant being y prime where a is dropped I have just be prime z prime with me. I have to check the validity of that new cube this is the representation I have to intersect it against all the cubes in the offset, and as it turns out what happens if you intersect this with the offset b c b prime conflicts with the b.

So, you have a null in the second position at least right this is b prime and this is b c, let us say that will lead to a null for the intersection with the first cube of the offset leads to a null. So, we are intersection with the second cube would lead to null in the third place because there is a c here and there is a c prime here. So, I have here a cube b prime c prime that has a null intersection with everything in the offset, so conclusion is it is safe to do that expansion. So, y prime is n since it is let us push further with respect to the expansion and try to drop something else now let us try to drop b here, that if you drop b then that 1 0 becomes 1 1 and I get the second expansion is just c prime.

If you take c prime here you will notice that that also leads to a null intersection with the offset because it conflicts with both of those they are in c form and since I have a c prime that is also a conflict and therefore, this expansion is also safe, it is a valid expansion, after c what? What's the next expansion as long as it is valid, we will keep expanding right now if this 1 is valid. So, what's the next expansion there is only 1 0 remaining you try to get rid of that 0 change that 0 to a 1 what is the resulting function.

Student: 1 sir; 1 everywhere.

Yeah, it is a 1 everywhere it is a.

Student: (Refer Time: 44:37).

Yeah, everything is do not care, so that one of; obviously, has a non-null intersection with the offset. So, that is an invalid expansion ok, similarly I can take one of the others if I try to expand w, then that would lead to something. So, that is my overall strategy, so

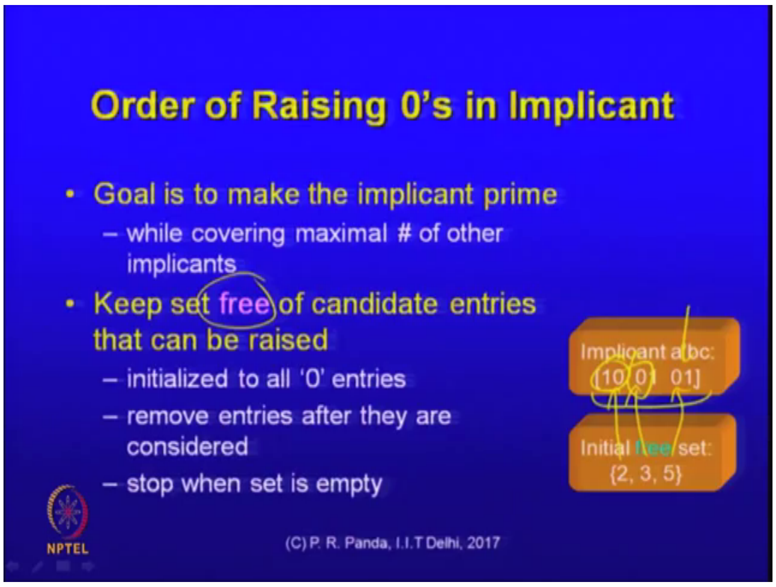
I have a table of these implicants and I know what the encoding is what the representation is my expansion would lead to one of those 0 bits becoming 1 and I do that intersection with the offset. In the process it turns out that x and z are covered by that expanded y since I had a c prime here, c prime is what I ultimately pick up.

So, I pick up two things 1 is the second expansion of y which I have indicated as y double prime, first expansion was b prime c prime second expansion was c prime that is what I would keep similarly w is what I would also keep that is my optimized set of cubes ok. In the process of that expansion it turned out that x and z are already covered I can drop them you can see they are covered because.

Student: Take an intersection.

Yeah, you take an intersection of that with this you actually get that term itself that implicant itself intersection with voidable prime will lead to since this is do not care whatever is there at a that will appear and this one is not conflicting therefore, whatever is there in that implicant will also happen. So, that is the expansion process and implementation would look like this.

(Refer Slide Time: 46:32)



Order of Raising 0's in Implicant

- Goal is to make the implicant prime
  - while covering maximal # of other implicants
- Keep set free of candidate entries that can be raised
  - initialized to all '0' entries
  - remove entries after they are considered
  - stop when set is empty

Implicant a bc: [10 01 01]

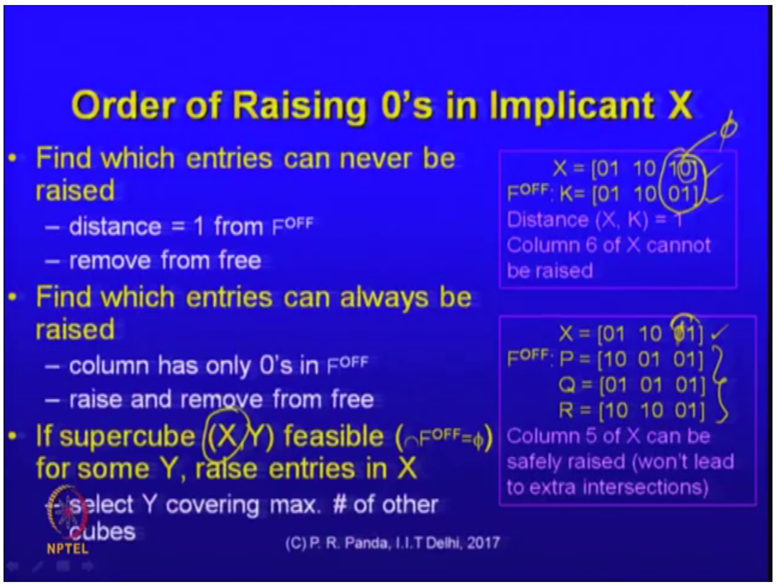
Initial free set: {2, 3, 5}

(C) P. R. Panda, I.I.T Delhi, 2017

Second would be the order of raising the zeros in an implicant the goal of course, is to make that implicant prime, but just because there are many zeros in our implicant it does not min that we can raise all of those zeros to 1 right. So, some subset has to be chosen and the number of subsets of course, is large. So, some heuristic is also needed here for choosing the order in which the zeros will be raised because at some point you have to stop you start with this perhaps it is ok. The next 1 it is invalid and therefore, you will have to stop on the other hand if you had chosen the third 1 to start with you would have arrived at a different expanded cube, so that order is essential.

So, let me keep a set free of the candidate entries that can be raised initially in this example there would be three this just refers to the position number 2 3 and 5, all of this are an initial free set ok, but they are candidates that does not min that all of them can be raised we have to check the validity to see which ones can be raised anyway it is initialized to those 0 entries in the process of raising we would drop some elements from that set of course, if you managed to raise all the zeros to ones then that set is empty.

(Refer Slide Time: 48:16)



Order of Raising 0's in Implicant X

- Find which entries can never be raised
  - distance = 1 from F^{OFF}
  - remove from free
- Find which entries can always be raised
  - column has only 0's in F^{OFF}
  - raise and remove from free
- If supercube (X,Y) feasible ($\cap F^{OFF} = \phi$) for some Y, raise entries in X
  - select Y covering max. # of other cubes

X = [01 10 10]
F^{OFF}: K= [01 10 01]
Distance (X, K) = 1
Column 6 of X cannot be raised

X = [01 10 01]
F^{OFF}: P = [10 01 01]
Q = [01 01 01]
R = [10 10 01]
Column 5 of X can be safely raised (won't lead to extra intersections)

(C) P. R. Panda, I.I.T Delhi, 2017

Some pruning of the search space can also be done here to reduce the number of choices that we have for example, first we can find entries that can never be raised. What entries can never be raised, suppose my offset was this consisting of 1 cube and this is the cube under consideration that we are trying to expand, this cube has a distance of 1 with the offset right it is different in the third position. So, the distance is 1 remember if there is a do not care somewhere then that does not count towards the distance right.

So, there is when there is a conflict a versus a prime that is when there is a distance you are counting the distance as the number of voids that are there in the intersection this is essentially 1 void. If the distance is 1 from the offset then you cannot raise it because that would lead to a non-null intersection with the offset, it is different at this point right it is essentially it is different because of that entry and if that entry is raised to a 1 currently it is the null intersection, but it will lead to a non-null intersection that is not allowed as part of the expansion process.

So, if those implicants are at a distance of 1 then those corresponding entries because of which that distance is 1 are the ones that we can throw out we need not consider those entries. Similarly, you could argue that there are some entries that can always be raised let us see an example of this, that is my cube under consideration and this is my offset and we are considering that position this is 0, this column has only zeros in the offset .

Since, it is only zeros here too I have a 0 1 there is no harm in raising this to a 1 it will not lead to any further intersection with the offset there already is some intersection whatever is there if there is a distance it came about because of some other position not because of this position. So, it should be for me to raise this to a 1 since, all the other elements are 0 there all the elements in the offset are 0 there then I can say that it is to raise it. So, I can raise and remove them from the freest.

These are somewhat trivial sort of cases, but otherwise what we are looking at is if the super cube of x and y remember it is obtained by taking the bitwise or of the 2 cubes, if it is feasible feasible means that there is no intersection with the offset right. So, where x is the implicant that we are considering y is one of the other implicants that we have with us, if it turns out that it is feasible for some y you raise the corresponding entries in x this is right.

As long as we are not intersecting the offset it is to raise it in whatever direction. Specifically, it is to raise it in a direction where a super cube is formed with one of the other implicants y is the super cube relevant here. So, I had some terms these are the offset terms ok, let us say this is 1 cube and this is some other cube the super cube of those 2 cubes it turns out is in this space that is this is the super cube it is there in a space that does not intersect with anything in the offset. So, we are saying there will prefer to expand along a direction that results in that super cube.

Student: Super cube.

Why would we prefer that?

Student: Sir, logic would become smaller the larger the cube that we are taking l.

Right, the logic does because I remember our ultimate objective is to not merely reduce the size of them, but the number of implicants that I am covering choosing the super cube leads to what possibility, this was my x this was my y the point of choosing the super cube is that I could drop that other guy.

Student: (Refer Time: 53:49).

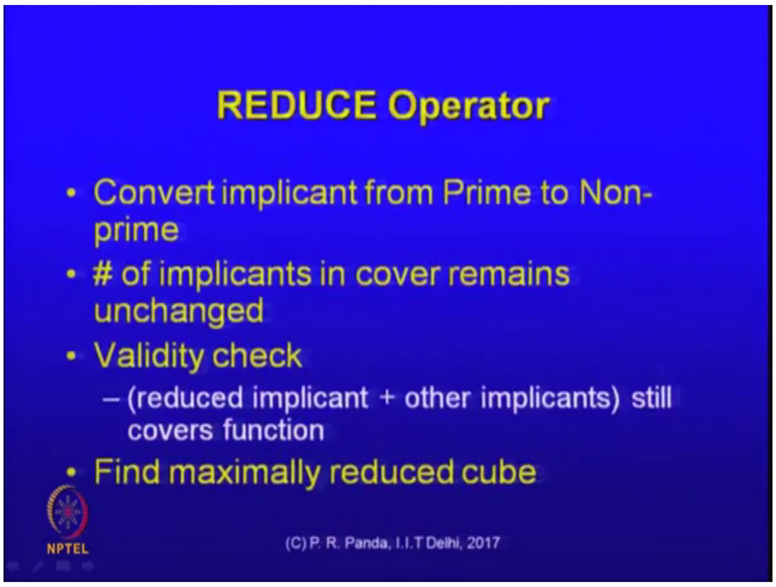
I am considering x this is the implicant that is under consideration we want to see what directions to expand it in, it could be expanded in different directions, but the profitable direction of expansion is 1 in which I am able to ultimately drop one of the other implicants, super cube is a nice thing that helps us here because if I choose the super cube, it means that one of these become redundant the both of those become redundant, but one of course, is the result of the bigger one is the result the super cube is a result of expanding one of them, but in the process I would have dropped one of the others.

I may have a choice of expanding in some other direction which might have lead to some other logic that is less profitable because, it does not result in that containment relationship that we want to move towards. So, that is the operation and among the different choices that we might have x is the one under consideration here, and the super cubed y could be taken super cubed z could be taken it is possible that when you take the super cube with z. In fact, or maybe it intersects with one of the others therefore, it is not a valid super cube, right.

So, you would select that y that covers in the process the maximum number of other cubes, yeah it is it is also possible that not only is y covered as it is y is covered in an obvious way when you expand x into the super cube of x and y, but it is also possible in this process there is some other cube that gets covered in that super cube remember super cube consists of not merely the union of 2, but some other minterms also.

That is really some essential ways if I want to distil the logic these are the things to take care of in determining the order in which the zeros can be raised in an implicit. So, two questions need to be answered for expansion one is the order in which the implicants are going to be processed, and other is for the implicant that is being processed the order in which the zeros would be raised to 1.

(Refer Slide Time: 56:12)



REDUCE Operator

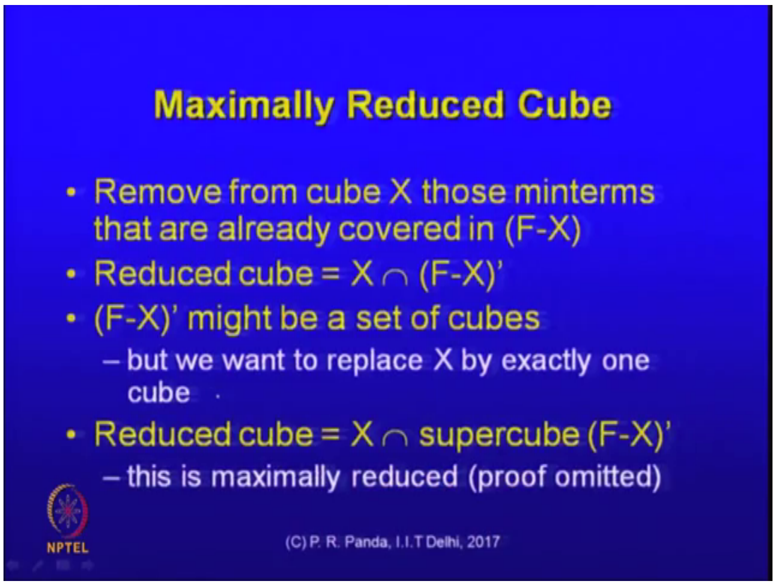
- Convert implicant from Prime to Non-prime
- # of implicants in cover remains unchanged
- Validity check
  - (reduced implicant + other implicants) still covers function
- Find maximally reduced cube

(C) P. R. Panda, I.I.T Delhi, 2017

Moving on to the reduce operator idea is now to convert a prime implicant into a non-prime implicant right the whole point of reduction is that it is no longer a prime implicant right it violates the definition of a prime implicant; however, the number of implicants in the cover still remains unchanged. In fact, even because of expansion the number is not changing it is just that redundancies are created in the process of expansion we will drop them as part of a different operation perhaps.

So, the number is not changing and a validity check needs to be made just like in the expansion case I made the validity check that we are not intersecting the offset. Here the validity check is the other way around we have to ensure that as a result of the reduction what condition still holds, the onset points are all still covered as a result of the reduction right.

(Refer Slide Time: 57:25)



Maximally Reduced Cube

- Remove from cube X those minterms that are already covered in (F-X)
- Reduced cube = X ∩ (F-X)'
- (F-X)' might be a set of cubes
  - but we want to replace X by exactly one cube
- Reduced cube = X ∩ supercube (F-X)'
  - this is maximally reduced (proof omitted)

(C) P. R. Panda, I.I.T Delhi, 2017

So, I want to find a maximally reduced cube here too I need this some heuristics, but let us define this maximally reduced cube first. So, I have some cubes let me represent this as a set. So, I have my x that is 1 cube and I may have some other implicant. So, this is a cube y and maybe some other cube these are my f that is my function.

First of all let me generate f minus, x which consists of essentially the remaining implicants the remaining cubes if you just remove that cube from there. Here is what I can do f minus x is it consists of this much right that and this together. So, that is my f minus x I take the complement of f minus x compliment with respect to set means what, what is the complement in this picture of f minus x each is seen as a set x is a set y is a set z is a set and actually formally.

Student: Why is x.

Complement is defined in terms of a universal set everything other than that what's complement of f minus x.

Student: 1 minus 1 minus z.

So, if you just remove the y and z all these others form the complement that is my compliment. So, f minus x prime is this. So, what we are saying is that reduced cube that we want is the intersection of x with f minus x compliment which gives us what .

Student: The centre part.

Yeah, this part of x that is not overlapping with the others, this is clear that that would be the maximally reduced cube I cannot reduce it any further because then I would drop some things from the function, but these parts of x that are overlapping with y and z, I can afford to drop from my reduced cube because they are already included in y and z anyway. So, that is all that we are saying x intersection f minus x prime is what is my reduced cube ok.

This is fine as the definition, but the problem is that f minus x prime that reduced cube this is just a set of minterms right that might not result in 1 cube it might actually become multiple cubes, that you can see that if you have let us say these were my original x. And that part was covered by y this part got covered by z then my maximally reduced cube consists of this, but this is not 1 cube this is actually a set of minterms.

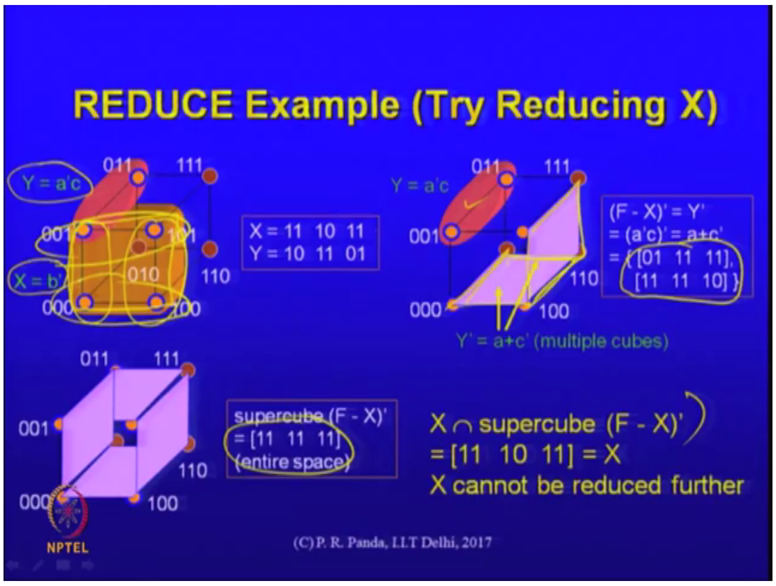
So, I cannot possibly have that whole thing, the problem is we want to replace x by exactly 1 cube that is part of the reduce operation in the process of doing the reduction we should not increase the number of cubes that are there that is that is through all of these operations we try to make sure that we are moving in a direction of reduced number of cubes. So, we do not want to increase the number of cubes.

What happened is we all had individual cubes that were all nicely structured, but the complement operation interfered with the single cube property it resulted in a possible multiple number of cubes, intersection is fine actually if you take the intersection of 2 cubes according to our definition of intersection you get only 1 cube that you can see in the representation itself right there is only 1 cube that is there, but then since the complement resulted in a disjointed set of minterms that do not necessarily form a cube.

The intersection of x with that complement also may result in multiple cubes then the reduced cube that I am interested in is the super cube of f minus x x intersection the super cube of f minus x prime, that is the reduced cube that we are looking for just because f minus x prime is no longer possibly just 1 cube it may be more than 1 cube.

That this is actually the maximally reduced that will leave the proof of, but you can see that it must be safe to do it there is a larger cube is formed of that complemented set, and anyway we are intersecting that with x means that we are choosing only from what was there in x. So, it must be safe to do that, so that is the idea, so this is what theoretically we are looking for as the reduced cube.

(Refer Slide Time: 63:32)



REDUCE Example (Try Reducing X)

X = 11 10 11
Y = 10 11 01

(F - X)' = Y'
= (a'c)' = a+c'
= { [01 11 11],
[11 11 10] }

Y' = a+c' (multiple cubes)

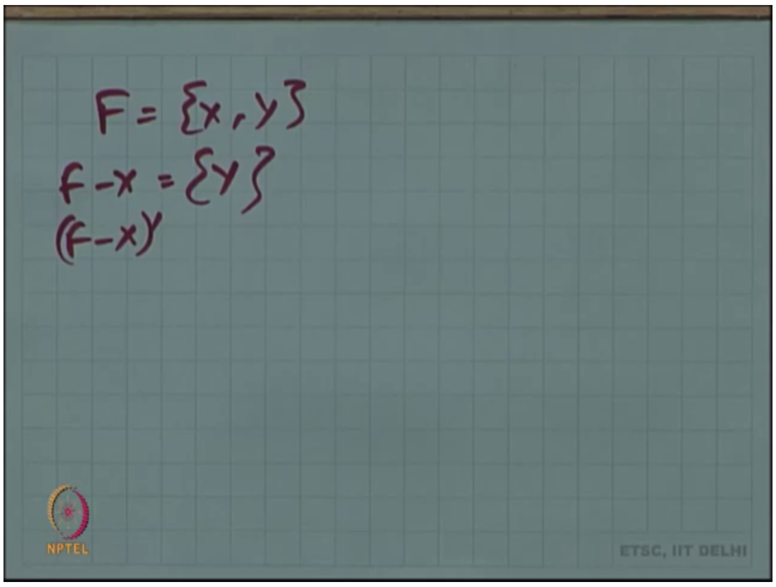
supercube (F - X)'
= [11 11 11]
(entire space)

X ∩ supercube (F - X)'
= [11 10 11] = X
X cannot be reduced further

(C) P. R. Panda, I.I.T Delhi, 2017

Let us try to do it too quickly on an example and it is associated representation, suppose I start with this function in which I have 2 cubes 1 is b prime this is the square and the other is a prime c it is representation is this. So, for x this is b prime and for y it is a prime c. So, that is my initial function I would like to reduce something let us try to reduce x. What is f minus x in this example, f consists of what is f here is the set of all the cubes that we start off with; so, x and y that is the set of all the cubes, so f equals just x and y what is f minus x.

(Refer Slide Time: 64:27)



$F = \{x, y\}$
$f - x = \{y\}$
$(f - x)'$

NPTEL
ETSC, IIT DELHI

Student: y.

Y; just y you remove that element, what remains c x that is f minus x what is the prime of f minus x prime is defined in terms of the universal set all the minterms that are not there in y that is the prime. So, you see that this may lead to multiple cubes because why was this is my y.

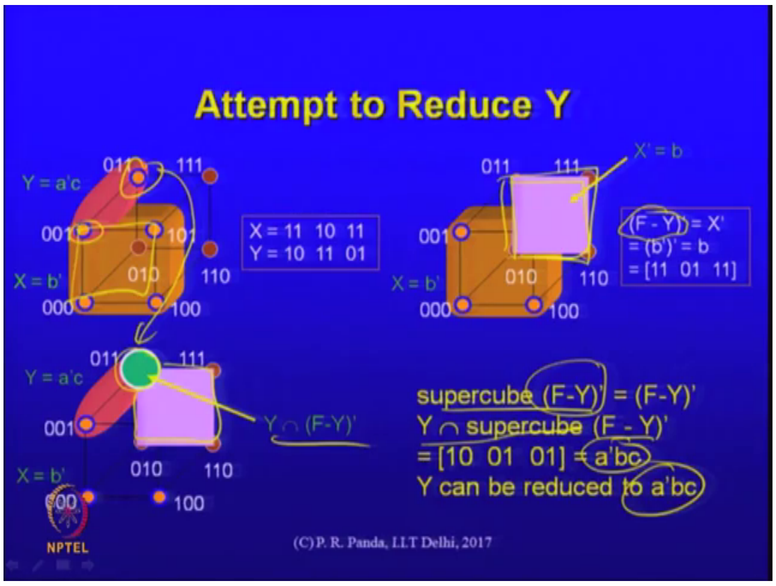
So, the set of all minterms that exclude ys is essentially the set of these 6, all of these 6 minterms that is f minus x prime. In fact, that results in 2 cubes here you can see one is this square that will cover those four and the other is this square, this is where the number of cubes is increasing in that complement operation ok. Then what is the super cube of those 6 terms it is essentially of these 2 cubes, from the representation you can see super cube of those 2 cubes is obtained by taking the bitwise or which is essentially the entire.

Student: Sum of.

Space everything, so that is the super cube now when you take the x intersection super cube of f minus x that is my definition of the maximally reduced x that I can afford. So, that when you take that intersection of x with or everything do not care you get x back, which means that x cannot be reduced further which x has to be there you cannot reduce x further, that should be clear in the following way x is the square, right.

If you reduce it what happens if you choose along this direction to either pick up these two or those two that would be a reduced cube, both of them would be illegal because in the process we would be dropping some minterms from the onset or it does not matter even if you pick up these two or these two, because all four are there in the onset you cannot really afford to reduce it in any direction.

(Refer Slide Time: 67:14)


Attempt to Reduce Y

X = 11 10 11
Y = 10 11 01

$(F-Y)' = X'$
$= (b')' = b$
$= [11\ 01\ 11]$

supercube $(F-Y)' = (F-Y)'$
$Y \cap$ supercube $(F-Y)'$
$= [10\ 01\ 01] = a'bc$
Y can be reduced to a'bc

(C) P. R. Panda, I.I.T Delhi, 2017

What about reducing y; it seems that there is a possibility for us to do something about it, but formally we have to take f minus y prime what is f minus y.

Student: X.

F minus y is just x right f consists of just x and y, so that is x. So, x prime for us since x is the set of those four points x prime is the set of those other four points. Now this actually has a reasonable overlap with this when you take the intersection of y with this set now you do get something. So, super cube of f minus y prime is what, f minus y prime is the set of those 4 minterms super cube of those four main terms each taken as a cube is what you end up with 1 cube only right this is an example where the super cube results in just 1 cube.

So, that is what it is and y intersection with this is just a reduced version of the y, you can see that this part of y is overlapping with x you need not have it in the reduced form, that part definitely needs to be there because there is no other prime implicant that is covering it, but that actually comes out here in our computation of y intersection f minus y prime. So, the result is that y could be reduced for us to a smaller region in a way that between x and y we are still covering all the elements in the onset, and we are not leaving anything out these are little further insight into a couple of those operations I have omitted irredundant the three operations like expand and.

Student: Reduce.

Reduce and irredundant that I have omitted, but the set of operations and the kind of things that you do is again similar.