

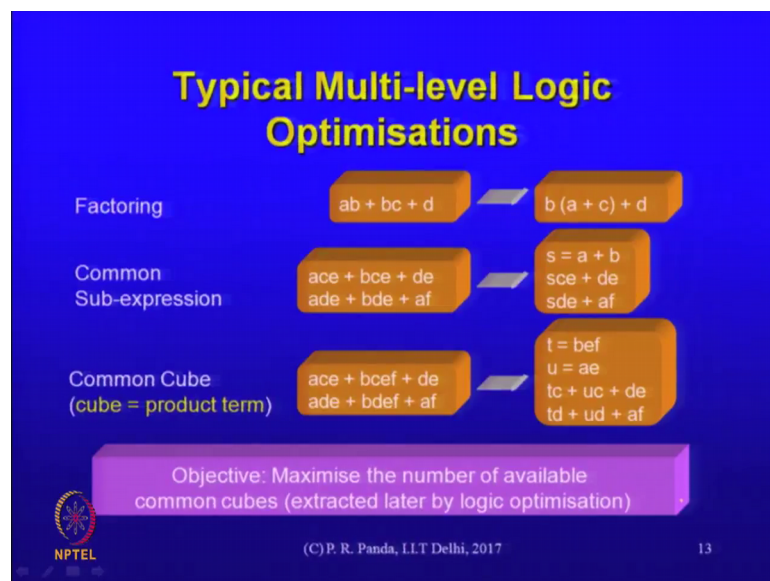
Synthesis of Digital Systems
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 17

Finite State Machine Synthesis: Identifying Common Cubes & Graph Embedding

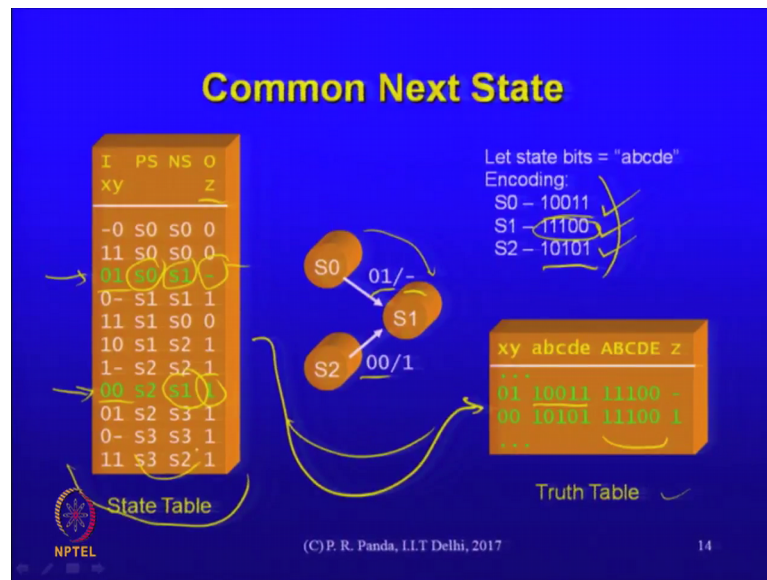
Ok so, we identified these basic optimizations that a multi-level logic minimizer would perform.

(Refer Slide Time: 00:31)



And our objective is that due to the encoding that we select, we somehow enhance the availability of these opportunities to the multi-level logic optimizer. And again the target here is multi-level logic and some of the arguments that are made later. Have this in mind for example, if the target were different, if it were 2 level logic then the arguments would not hold the specific arguments would not hold ok.

(Refer Slide Time: 01:08)



Let us go through some of the basic ideas of this particular strategy; this is S strategy that we are studying in more detail.

Primarily because, there are some ideas there that are generic enough that help us formulate the problem that help us. Also understand the relationship between what happens at an early stage a problem and its solutions such as this S1 and its interconnection with later stage and strategy, later step of a logic minimization. Happens later ideally one should be aware of what goes on there at the time that we make decisions here, but due to the sequential nature of the flow we have to make certain assumptions rather than running logic optimization at this stage. We make certain assumptions and estimations about what will happen later ok.

Let us study in a little more detail some patterns that we can identify in the state machine that is given to us and in terms of that we would formulate the solution to the encoding problem. So, this is this state machine that is given the table is an alternative way of representing it. And what we are pointing out here is a couple of rows this S1 and this S1. Ok, that corresponds to a subset of the FSM that looks like that; you have a common next state with 2 different present states S0 and S2 right that is the state diagram part of the state diagram.

So, that is the pattern that we have identified, there is a common next state S1 and a 2 different present states S0 and S2. These are of course, the inputs x and y are the inputs

to the FSM. So, they show up here that is the condition under which you would make the transition from state S0 to state S1. And what you have here is the output, there is a single output here z and here that output is do not care, but here that output is 1. So, this is of course, an equivalent representation of just those 2 rows, each row corresponding to an edge in the FSM a state transition.

So, we identified this pattern on the basis that there is a common next state and there are 2 different present states. What is the implication of this let us study by going all the way to the Boolean expressions and visualizing what the gate structure might be that would help us implement this ultimately. Now, if I take this state table and do an encoding let us use this encoding here using 5 bits just for illustration, number of bits of course, is an orthogonal decision and you can assume that maybe what we have identified here is only part of the FSM. There may be many other states, that is why we are using a 5 bit encoding for the FSM states.

So, let us assume this kind of an example encoding is made, that is the encoding for S0, that is the encoding for S1, that is the encoding for S2. And when you do that encoding what you end up with is a state table is converted into a truth table that looks like this wherever you have S0 that gets replaced by the encoding for S0 that is the encoding right. So, that gets replaced there S1 is replaced by this and S2 is replaced by this. So, this truth table of course, is a function of what a encodings we have chosen for the individual states.

Any questions on this? This is an elementary implication of a an example encoding that we would perform. Now let us study what happens as a result of the encoding. Of course, the a state table is converted into a truth table; that means, that all the states the symbolic states we have here are assigned their respective encodings and we have a truth table here ok.

(Refer Slide Time: 06:00)

Logic Equations after State Assignment

The slide contains the following components:

- Truth Table:**

I	PS	NS	O
xy	z		
...			
01	s0	s1	-
00	s2	s1	1
...			
- Block Diagram:**
 - State Reg:** Receives inputs ABCDE and produces outputs abcde.
 - Comb. Logic:** Receives inputs xy and produces output z.
- Logic Equations:**

$$A = x'yab'c'd'e + x'y'ab'cd'e + \dots$$

$$B = x'yab'c'd'e + x'y'ab'cd'e + \dots$$

$$C = x'yab'c'd'e + x'y'ab'cd'e + \dots$$

$$D = \dots$$

$$E = \dots$$

$$z = x'y'ab'cd'e + \dots$$

Let us continue to the Boolean expressions that would implement the truth table. So, the truth table has these on the output right, this is the input section and that is the output section of that truth table. These outputs correspond to what?

Student: Next.

The next state so, those are essentially these A, B, C, D, E value. So, we are trying to build the combinational logic for this, part of that logic is the output, that is this and the rest of the output here essentially is going into the state register. So, the truth table of course, captures both of these outputs, one is the next state signals, the other is the output signals of the FSM. So, let us build the logic, I have to build an expression for all of these 6 outputs right one for a year questions.

Student: So, the state register will be giving the previous or a present state and the combination was a output will give the next state.

Right.

Student: So, small a, b, c, d, e should be after the second state (Refer Time: 07:15) capital A, B, C, D, E should be before the state register.

These are my outputs from the combinational logic right. So, these are the D inputs of the flip flop, those are the cube outputs of the flip flops. Yeah, I did not draw the

direction of the arrows, but that is what it is. So, capital A, B, C, D, E are the outputs of the combinational logic they feed into the state register ok.

So, now, what I need to do is for each of these columns output columns I need to pick up all of those rows where that output is 1 and build the expression or the other way around I can pick up all those rows where it is 0 and build an expression and invert it by the way both are equivalent.

Ah so, let us just take all the ones that are there and form the logic. So, A will be equal to this expression where does this come from plus this? So, these are 2 expressions there 2 terms in the expression for A, where do they come from? And why do we have these 2? One should correspond to that, one of these terms should correspond to that row being one for a which means that we will pick up the corresponding input here. First let us look at the x y so, x y is 0, 1 so, that is my x prime y abcde we had 0 there this is the encoding for 0.

So, I have b prime c prime and A, D, E that is the expression. So, that is what I have here so, that is my expression corresponding to these inputs. Then the other one would correspond to picking up that expression, that is what I have that. For B what would I have it is the same as this as the why is it the same? The expressions for A and B are identical so far why are they identical?

Student: Because both A and B are one together.

Both are one together at least as far as these 2 rows are concerned they are indistinguishable so, it should be the same expression. Same for C, D and E, I have nothing here. Why do I have nothing? Because these are zeros right so, my strategy was that in building up this Boolean expression. I will pick up all of those rows where that output is 1 at A so, where it is 0, I just ignore it. The at least these 2 rows do not contribute anything to D and E.

What about z? There is also an output so, I have to build the logic for Z. for Z, I have 1, but the other one I do not have anything. What I have here is a do not care, a do not care of course, means that you could treat it as 1 or the other. For now let us just leave that if it is do not care which means that you treat do not care as A 0. that is just for illustration purposes we do not of course, actually treat do not care as A 0, but this is essentially

what is corresponding to the second row same input conditions. So, that is how I would build up my Boolean expressions for implementing A, B, C, D, E and z all the outputs.

So, that is the logic that that combinational logic block is supposed to implement. Of course, as of now I have only specified very explicitly what those expressions are, this is not an optimized form of that expression, but it is just like the specification of a Boolean function so, that we start from here. Any questions about the way we form this? This is of course, an important first step in any further arguments that we make.

(Refer Slide Time: 11:20)

Identifying Common Cubes

$$A = x'y'ab'c'd + x'y'ab'cd'e + \dots$$

$$B = x'y'abc'd + x'y'abc'd'e + \dots$$

$$C = x'y'ab'c'd + x'y'ab'cd'e + \dots$$

$$D = \dots$$

$$E = \dots$$

$$z = x'y'ab'cd'e + \dots$$

$$s = ab'e$$

$$A = x'yc'd + x'y'cd'e + \dots$$

$$B = x'yc'd + x'y'cd'e + \dots$$

$$C = x'yc'd + x'y'cd'e + \dots$$

$$D = \dots$$

$$E = \dots$$

$$z = x'y'ab'cd'e + \dots$$

Encoding (N bits):
 $S0 = 10011$
 $S1 = 11100$
 $S2 = 10101$

Hamming Distance $H(S0, S2) = 2$
 Lines for A, B, C have common cube of size $N-H = 5-2 = 3$
 Similar observation for O/Ps

NPTEL (C) P. R. Panda, IIT Delhi, 2017 16

Those were my set of expressions and let us get to identifying optimization opportunities in those expressions. optimization opportunities for us right now are the process of identifying common cubes because, we said that that is what we can concentrate on, it is easier than finding general common sub expressions. Common cubes if I look at that expression for A, I see that there is a common cube between that expression in this expression, that common cube is this much $ab' prime e$.

So, these 2 expressions are different in terms of sum of the variables, but they are common in terms of sum of the other variables are just trying to find out which ones they are common in terms of. And for now let us just restrict our attention to the state part of it, abcde part of it ok, not to the xy they can be treated similarly, but since this argument is about states, let us focus on the abcde.

So, C and D are different in these 2 expressions in terms of their appearance as normal form are complemented form, but a, b and e are the same. So, I can extract that portion that is highlighted in green as a common cube. And then I can express my A in terms of that common cube. you can see that this will lead us towards a better implementation at least in terms of gate size, gate count it should be better because I have extracted that common cube, I do not have to do that repeated ending instead I have done that ending only once. And that output S can be sent to various other expressions as we compose the multi-level logic implementation.

How did that happen though that common cube arose because of what? Remember that part was in the state table it was just a symbolic mention of S0 or S1 and S2 that is what it was. So, what is it that cause the appearance of a common cube? This was in part of the original specification.

Student: The encoding.

It is the choice of the encoding, remember we have a choice this is an example encoding that we have taken and due to that we could extract some common cube um. The 2 aspectS1 is the size of the common cube that size is 3 literals here other is the number of appearances of that common cube, both of them somehow influence the total area of the ultimate implementation ok. So, it is that encoding and if there was a common cube of size 3 it is due to what property of that encoding?.

Student: That 3 states were there and it could be encoded in 2 bits whereas, we were using 5 bits where 3 bits are redundant.

Oh remember what we have here is only part of a bigger FSM, it is certainly possible that I have a 25 states in the FSM and I would choose 5 bits in that case. So, the 5 was chosen just to illustrate the point it does not have to be fine, but even if it is a minimal encoding right then too you will find common cubes. The appearance of the common cube does not necessarily follow because of any redundancy in the encoding.

Student: So, the common next state is the theme. So, you would prefer to maximize the align the number of or rather when in when you have common state next state then make sure that the number of transition in the bits in the input states is minimum.

Student: Minimum (Refer Time: 15:52).

Student: So, that we can extract the maximum cube.

Why did we have a common cube of size 3 in this example? Between hamming distance between.

Student: (Refer Time: 16:02) the states.

Which 2 states?

Student: (Refer Time: 16:07) has different input (Refer Time: 16:08).

The 2.

Student: (Refer Time: 16:10) common output (Refer Time: 16:11).

Yeah the original state so, S_1 is the next state S_0 , between the 2 present states S_0 and S_2 there was a hamming distance of 2 therefore, the other 3 bits were the same and that is what led to the appearance of a common cube of size 3. there is nothing sacrosanct about that size 3 though if I wanted I could have got a size 4 how would I have got a size 4?

Student: It is need to change either of the (Refer Time: 16:40).

Changing the encoding of S_0 and S_2 in a way that they are distant by one unit could have had a common cube of size 5? For those states I cannot because that would violate the requirement that the states of course, be distinct in their encoding. So, that is what it is that the size 3 appears because of the choice of encoding our choice and from the fact that that hamming distance between those 2 encoding says 2, we can infer that the lines for A, B, C all of them will have a common cube of size whatever is the max. So, here the max is 5 minus 2 and that that is where the size 3 comes from ok.

Ah while this was the argument for the next state, a similar argument can also be made for the outputs. There is no difference right these are also output, the capital A, B, C are also outputs, the z is also an output there. So, why did we go through this argument? It helps us formalize at least something; intuitively what does it tell us? I can look for such occurrences of patterns in the finite state machine and if I find such occurrences then

what do I do? There are some things that I cannot fix, there are some things that I can fix that is up to me. What I cannot fix is the structure of the finite state machine that is given to me.

But what can I fix, the choice of the encoding is up to me and if it happened that this is all that was there or I find patterns like this then I can informally try to do what in terms of the encoding?.

Student: (Refer Time: 18:38) encoding (Refer Time: 18:39).

Try to keep these 2 states S0 and S2 as close as I can, that is really the intuition. There is the next step of how to quantify that intuition in some formal way. But, the idea is this informally that we are looking for some patterns and we already know that if those patterns exist then there are some implications with respect to possible optimization opportunities for the a logic minimizer later on. And what we have in our hands is the encoding and we can do something about the assignment of the encoding yes.

Student: Typically (Refer Time: 19:23) while writing even RTL's it is advised to keep the state as enum's in a sense so, that your RTL is readable.

Yes.

Student: So; that means, encoding is even the metal gate level synthesis where it is RTL too.

Yeah encoding.

Student: This point.

Is a more general problem this is being introduced here with respect to a finite state machine. But, as you can see all this logic in fact, has nothing to do with the finite state machine, this is a truth table there is a process of a symbolically translating that FSM into a concrete truth table. the idea of encoding is of course, more general.

You could have a encoding being defined for just combinational logic even if you did not have anything sequential, even if you did not have a finite state machine, but you just have an had an input and an output you have a truth table. But, in fact, you can for the

purpose of minimizing logic take some of those inputs and encode them as a some other values if it helps in some way the final logic that is being generated.

You could imagine conceptually an encoder at the in input stage, a decoder at the output stage and the intermediate logic is what you are trying to optimize maybe it makes sense to do such an encoding and decoding.

Student: (Refer Time: 20:49).

That is a separate argument so, we went through that right initially even before deciding on the encoding you have to decide the number of bits. And the we did talk about the 2 extremes, one is if you use minimum number of bits then what is the implication if you have a one hot kind of encoding then what happens ah.

Student: (Refer Time: 21:14) if they if we (Refer Time: 21:16) the number of states and according to that it does not come at the hot encoding because the number of the states are (Refer Time: 21:24).

No, no the number of states in that table is this is just 2 rows of that table, the table can be much bigger maybe there are 25 states in that in that table that is why you have 5 bits ah. So, it is a different reason this argument holds irrespective of what is the width of the encoding, I made it 5 just to illustrate the effect, but what I have here N minus H that is the one that is a general argument it applies for any N and any H .

Student: Sir for large FSM's?

Yeah.

Student: I mean because it is used even today at logic optimization in RTL2 gate level, this could become very intractable.

No, no we absolutely want a practical solution to this. This I have not indicated an algorithm.

Student: Right, but I have not just.

What do you mean it will become intractable since I did not tell you what the algorithm. We are still in step 0 of the algorithm and we have we are we are saying that if such a pattern was there.

(Refer Slide Time: 22:33)

Common Present State

I	PS	NS	O
xy			z
-0	s0	s0	0
11	s0	s0	0
01	s0	s1	-
0-	s1	s1	1
11	s1	s0	0
10	s1	s2	1
1-	s2	s2	1
00	s2	s1	1
01	s2	s3	1
0-	s3	s3	1
11	s3	s2	1

```

graph TD
    S1((S1)) -- "11/0" --> S0((S0))
    S1 -- "10/1" --> S2((S2))
    
```

Let state bits = "abcde"
 Encoding:
 S0 - 10011
 S1 - 11100
 S2 - 10101

xy	abcde	ABCDE	z
11	11100	10011	0
10	11100	10101	1

State Table

Truth Table

NPTEL

(C) P. R. Panda, IIT Delhi, 2017

17

Then you could do something about it. That is just one of the arguments; let us look at it the other way around. Identify a pattern where you have 2 transitions with a common present state that is these 2 grows that leads to a structure like that, S1 is the common present state, and S0 and S2 are the common next states.

For that example encoding there the corresponding to a truth table rows will be something like this right S1 is encoded into this S0 is encoded here, S2 is encoded here. That is the encoding, what optimization possibilities we see there.

(Refer Slide Time: 23:30)

Common Cubes after State Assignment

I	PS	NS	O
xy			Z
...			
11	s1	s0	0
10	s1	s2	1
...			

$$A = xy'abcd'e + xy'abcd'e + \dots$$

$$B = \dots$$

$$C = xy'abcd'e + \dots$$

$$D = xy'abcd'e + \dots$$

$$E = xy'abcd'e + xy'abcd'e + \dots$$

$$z = xy'abcd'e + \dots$$

S0
11/0

S1
10/1

S2

xy	abcde	ABCDE	Z
...			
11	11100	10001	0
10	11100	10111	1
...			

$$A = xy' + xy's + \dots$$

$$B = \dots$$

$$C = xy's + \dots$$

$$D = xy's + \dots$$

$$E = xy's + xy's + \dots$$

$$z = xy's + \dots$$

Encoding (N bits):

S0 - 10011

S1 - 11100

S2 - 10101

Hamming Distance (S0, S2) = H

Up to N-H = 3 Next State Lines (A, E in this example) have common cube of size N = 5 (irrespective of Encoding for S1)

Similar observation for I/Ps

Let us go through that same process of generating the Boolean expressions. So, from this I again construct the expressions for A, B, C, D, E, z all of these um. The expression for A has what terms? It has this term and this term. Expression for B has nothing again of course, that is because of the choice of the encoding both are zeros for C so, these are 2 different states right.

So, the encodings are different 0 in one case and one in the other. So, I would pick up the expression corresponding to that row for D similarly I would pick up the expression corresponding to this row. For E, I would have the same thing that I had for A. Similarly for z, I would pick up the same thing that I had for C. So, I have this set of expressions, what can we say about the availability of common cubes in this set of expressions. The transitions depend on the inputs, this is a common present state of course,. So, if I look at these 2 expressions in fact, focusing our attention on A, B, C, D, E only this entire expression is common. Why is it common? Is it because of anything we did in the encoding?

Student: no sir.

No, why is it common common for all the 5? Because it is the same present state it does not matter what the encoding I had given that is going to be common, that is because of the pattern that I am looking for that. Since it is the present state is common, I would have a common cube of size 5 no matter what the encoding.

So, generalizing if I had N bits then if the hamming distance between the 2 next states S_0 and S_2 if that was H , then what can I say now about the availability of a of common cubes. So, I was able to extract this out, that is good it is size is even bigger it is the size N common cube. It occurs in some places if you look at the individual expressions for the outputs some of them it is certainly occurs for A, it occurs for E. Does not occur for B, also within the C's expression and D's expression we did not find it, but we did find it within A's expression and E's expression.

So, what is that attributed to? Some of the next state lines would have a common cube of size N or size 5 that is what it is how many would they be. So, 2 things we are looking for one is the size of the common cube, other is number of occurrences of the common cube. Size is 5 in this case this is not dependent on the encoding, but what about the number of occurrences? Viewed independently, view the logic for A independently, logic for B independently and so, on.

Ah there is an opportunity for A, there is an opportunity at E, but there is no opportunity here bit for B, C, D. This happened because of what? And in general how many next state lines out of the 5 next state lines we expect a common cube in?

Student: The number of ones in the next state including.

The number of ones. My question is view this logic differently, all of these logics differently, logic for A differently, logic for B, for each of them we would like to identify is there a common cube of size 5. So, for A it is there, but for B it is not there, for C also C and D also it is not there. Within the logic of C and D it is not there ah, but for E it is there. So, how can I characterize the number of appearances of the common cube within the respective expressions?

Student: (Refer Time: 27:49) distance between repetition of of basically repetition of my present state. So, with this input combination these are my present states and with next input combination these are my present states so, whenever my presents.

This is a pattern in which the present state is common.

Student: Yes.

So, in both the rows the present state is the same, what is different is the next state.

Student: Yes.

Student: So, the instructs (Refer Time: 28:19) from the A to D (Refer Time: 28:22) is common right from the A, B, C, D, E in the first (Refer Time: 28:25) 1, 1 (Refer Time: 28:26).

There is 1 1 right so, how do I in general characterize it? Number of a common cubes that would be there.

Student: For the same present state the number of inputs that will change (Refer Time: 28:40).

Yeah so, this pattern is the same present state same present state is leading to a common cube of size 5 right, but what is leading to the appearance of 2 common.

Student: (Refer Time: 28:54) present state and the next state.

Cubes the distance between.

Student: Number of.

Present and next states.

Student: Number of inputs (Refer Time: 28:59).

What difference it makes the distance between present and next states? Somehow something that is the same between A and D right, what is the same?

Student: So, the A, B is same and E is same and only 2 bits are changing so, somehow that 2 is still related to it, I cannot explain it in why.

It has to do with the way we are encoding the next states, these 2 states have something in common that is showing up in those expressions there. What do they have in common? See S1 it does not matter what you assign that is showing up here right and we are saying anyway that is the expression does not matter what the encoding of S1 is. You will find a common cube of size 5 and what that the actual value is abcd prime a prime that is derived from the 1 1 1 0 0 that was chosen for S1.

If you had chosen a different encoding for S1, you still have a common cube of size 5, but that expression would be different.

Student: (Refer Time: 30:05).

Yeah.

Student: (Refer Time: 30:06) it will add some clarity I hope. So, in previous case we are looking at vertical vertical common cube right, here are we looking for a horizontal common cube?

Well why do not we look at this arguments are certainly different, they are not exactly in the same we are looking for a common cube here within the expressions.

Student: (Refer Time: 30:26) horizontal.

Right, but actually the previous one was also not very different. Here too we were looking that it exists across the lines is a different matter, but there 2 the common cube did exist.

Student: But here we (Refer Time: 30:41).

For the output line.

Student: (Refer Time: 30:43) that is the (Refer Time: 30:44) distance between input pattern.

So, why did we have 2? First of all where the bit positions at which the bits are different corresponding to S0 and S2, those correspond to the C and D, where anywhere there is no common cube right. So, we are restricted to the other 3 positions that are actually common between the 2 encodings. Now, but even in those 3 it turns out that one of them is a 0 and that does not contribute to a common expression there because we are actually ignoring it.

So, the common part is 1 0 1 the, that is A, B and E, but since B is a 0 then it does not really contribute towards anything. what is actually contributing is that part that is common between the 2 next states encodings that actually have a 1 in them. So, what is common between S0 and S2? It is that 1, 0 these things are not common and a 1 right.

What is highlighted in pink that is what is common between S0 and S2. You can see that the expression for A so, these are of course, A, B and E this commonality is showing up in these expressions that we have built. This being one in both the cases is actually showing up in that expression that S repeating here.

Similarly, this being one in both of these cases S0 and S1 that is showing up in the S repeating here there C and D not being common is showing up in those expressions not repeating there, that D had a potential to be common if we had encoded this as a 1 then that would show up here a common cube it show up, but actually since it is a 0 we are not even picking anything. So, there is nothing here.

So, first I can identify the hamming distance between them and get my N minus H . So, these are the 3 potential lines where a common cube could exist, but at the lines where it will actually exist would depend on which ones I have given a 1 there, the ones that I have given a 0 those would not show. That is what the logic is, this is in some ways a different argument from that common next state logic that is why we went through it separately ok.

(Refer Slide Time: 33:41)

Strategy

- For an FSM, we can identify many such relationships
 - distance of codes vs. size/number of common cubes
- Estimate reduction in literals (or “gain”) if two states are assigned close codes
- Assign codes that maximise the number/size of common cubes
- Assume that logic optimiser will find and exploit the common cubes

NPTEL (C) P. R. Panda, IIT Delhi, 2017 19

So, these are my patterns, for an FSM we can identify many such relationships. So, this is just 2 rows that we just looked at out of a bigger FSM and I can use whatever algorithm to traverse the FSM and look for patterns like this as a common present state or common next state. And through it and quantify in some way the distance of the codes

versus the size of the common cubes and the number of the common cubes that I exist, this is what we have in our hands. The encoding can be made in a way that the distance is minimized or maximized in a way that the size or the number of common cubes.

The availability of those is maximized. So, the strategy would be as follows, you try to estimate the reduction in the literals that is what we call the gain, if 2 states are assigned close codes. both of those patterns that we identified in the process we inferred that a particular pair of states should have close codes, that is informally what we want. And there is a little bit of a detailed quantification of a how do you prioritize them? Which is higher priority? Which is lower priority? That priority is identified first and then we will come up with a strategy that takes that priority into consideration.

But, you estimate that reduction if 2 states are assigned close codes and by identifying those common next states or common presence states we can try and put numbers there saying there is a greater chance of a literals being reduced if these 2 states were close. So, that when there is a competition for the actual encodings, priority can be given to making the those particular pairs close that are more worth it with respect to how we have estimated the reduction in literals.


So, I would like to assign codes that maximize the number or the size of the common cubes. And of course, the encoding process stops there, you assume that once those common cubes are available the subsequent logic optimizing step we will find and exploit those common cubes in the way that you expect it to. This is a basic requirement of any logic optimization tool.

You do not want to probably put too many dependencies between this higher step and the next step because then it becomes tool specific what you do here. But, the idea of a identifying and a common cube and exploiting it is something that is basic enough a requirement of a logic optimizer that it should work irrespective of which particular logic optimization algorithm is being used.

(Refer Slide Time: 36:42)

Formulation

- **Build Undirected Graph**
 - Nodes: states
 - Edge Weight: gain if states are assigned close codes
- **Assign an encoding that maximises total gain**

 (C) P. R. Panda, IIT Delhi, 2017 20

While that was the informal intuition, this has to be formalized in some way this has to be quantified so, that we can make a decision in the encoding. You build an undirected graph where the nodes correspond to the states and the edge weight is the gain if those 2 states are assigned close codes. So, that is the number that we will compute and we will put a number on each of those edge weights and the higher number which means that it is a more gain is there if they were close in their encoding. And you at the end after the graph is formed in a second stage you assign the encoding that maximizes the total gain.

So, there are 2 parts in the formulation, first is building that graph itself and the second is once you have the graph solving that graph in terms of an encoding, that would maximize a cost function in some way and that cost function here is the total gain for all the encodings together ok.

(Refer Slide Time: 37:46)

Building the Graph: a Fanout-oriented Algorithm

- PS's with the same NS and O/P get high weights
 - leads to close codes eventually
 - maximise size of common cubes ✓

ab	PS	NS	t
01	s0	s1	0
00	s2	s1	1

(C) P. R. Panda, IIT Delhi, 2017

So, let us look at the building of that graph itself Fanout just corresponds to one of those strategies there is a different strategy that is also covered in the paper called Fanin we will look at one of them in detail. Idea of course, is what we already said there are 2 present states with the same next state output of course, is there and they will get high weights the which means the edge connecting them, this is the graph that is being constructed. So, that edge will have high if we can find the pattern.

So, for example, that common next state pattern is there right. So, if that common next state pattern is there what is the logic? Logic is that S0 and S2 if they were minimally distant then I would maximize the appearance of those common cubes somewhere. So, the priority can be high. So, high priority here in terms of a higher annotated number this is an estimated gain for literals, that would lead to close codes eventually the second stage of the algorithm will make sure of that. And essentially what it represents is that the size of the common cube is maximized.

So, if I have something like this if this situation arises from the state table that I started off with then maybe in the later stage I choose an encoding like this where there is a large common part in the encoding size select for S0 and S2.

Student: (Refer Time: 39:26) multiple such common states.


(Refer Slide Time: 39:26)

Fanout: Constructing Output Sets

- If 2 PS's assert same o/p, common cube can be extracted
- Construct N_o different o/p sets
- Assign weight = #occurrences of the PS for same o/p

I	PS	NS	O
xy			z
-0	s0	s0	0
11	s0	s0	0
01	s0	s1	-
0-	s1	s1	1
11	s1	s0	0
10	s1	s2	1
1-	s2	s2	1
00	s1	s1	1
01	s2	s3	1
0-	s3	s3	1
11	s3	s2	1

OUTPUT SET_z
= {S1 (2), S2 (3), S3 (2)}



(C) P. R. Panda, IIT Delhi, 2017

22

Yeah.

Student: There can be a scenario where the priority of making 2 states same is same right so, the gain (Refer Time: 39:37).

Yeah so, that shows up in the quantification there is a graph that is being built and a quantification being made for every pair of a states. Some of those edge weights ultimately might turn out to be equal. We will quickly go through this quantification just giving the basic logic for doing so; you could quantify it in some other ways also. but, let us go through the same logic that is it is proposed by the authors of that paper. You construct some output sets if you had 3 outputs here, you would construct 3 sets. So, the z corresponds to the output there is only one output here so, we construct one of the sets in the following way.

If 2 present states assert the same output, then a common cube can be extracted for reasons that we saw earlier. You construct N_o if there were o outputs different output sets here o is equal to 1. So, you are constructing only 1 set. So, what you pick up is this all of these present states that assert that output means which lead to that output being 1, those are what we are picking up right. So, S1 is one of those states that lead to an output of 1, S2 is one of those states and S3 is one of those states. We not only keep those elements, but we also include the number of appearances of S1, number of appearances of S2, number of appearances of S3.

So, we are also counting the number of a times that present state occurs. why is that number essential? So, it is used later on essentially the more the number of times that present state is occurring the more the number of occurrences of that present state when you actually form the logic. Because, remember at least at the first step we are picking up that Boolean expression this Boolean every row we are picking up that expression right. So, that many times that expression will occur and some weight we are giving to a present states that occur more number of times.

(Refer Slide Time: 42:07)

Fanout: Constructing NS Sets

- If 2 PS's produce same NS, common cube can be extracted
 - intersection of the 2 PS encodings
- How many common cubes?
 - # 1's in Next State encoding $\leq N_b / 2$
- Construct N_s different NS sets
- Assign weight = occurrences of the PS for same NS

I	PS	NS	0
xy		z	
00	s0	s0	0
11	s0	s1	0
01	s0	s1	-
0-	s1	s1	1
11	s1	s0	0
10	s1	s2	1
1-	s2	s2	1
00	s2	s1	1
01	s2	s3	1
0-	s3	s3	1
11	s3	s2	1

$NS_SET_{s_0} = \{s_0(2), s_1(1)\}$

$NS_SET_{s_1} = \{s_0(1), s_1(1), s_2(1)\}$

$NS_SET_{s_2} = \{s_1(1), s_2(1), s_3(1)\}$

$NS_SET_{s_3} = \{s_2(1), s_3(1)\}$

NPTEL (C) P. R. Panda, IIT Delhi, 2017 23

Student: Sir on previous slide (Refer Time: 42:10).

Yeah.

Student: Should we also assess in this case we have only one output.

Yeah.

Student: A very simple case, but still in this case would it be faster for us to build a logic with 0 output and then negate it as necessary?

It could certainly be through all these illustrations we are making one assumption or the other, but everything that we are saying could well be inverted and you could consider the zeros at the different stages. And build a an inverter out of it yeah and ultimately whatever the output is taken that is inverted. The logic might still be the same. Here we

are not taking the decision though because remember this is a high level argument the actual logic optimizer is not yet there. You expect the logic optimizer to indeed evaluate the 2 and make the selection that is better.

Student: Even during encoding if I let us say in this state I had only two 0.

You could, this is an illustration only and all of this could well also be for example, note that they do not care is not resolved by us and at some point we just randomly assumed that it is it translates to 0 need not it could also be 1 and so, on. But the logic could also be used by looking at it the other way around and taking the inversion. You could still you reuse the same logic, but choose the zeros the complementary part of it instead ok.

So, that corresponded to the output. Now, let us look at the other part of the output which is the next state logic. Our idea was that if 2 is present states produce the same next state then a common cube can be extracted ok, that corresponds to the intersection of the 2 present state encodings that was our first pattern. How many common cubes are there? So, that corresponds to the number of ones in the a next state encoding that I do not know yet because, the actual encoding will happen later. But the still the quantification has to be made and the way that is approximated is assume that half the bits are going to be 1.

Where N_b is the size of the state register number of bits that you are choosing if that is N_b in our example there N_b was 5 width was 5 assumed that half of those bits are going to be 1. The other half are going to be 0 and that is something that will help us in the quantification how many cubes are there. 2^2 things are there right one is the size of the cube other is number of occurrences of that cube ok. So, now, you construct N_S different next state sets.

So, one for every state ok in each of them you capture those states which are present states with this S_0 being the next state that is what is the next state set. So, what is highlighted here in green are those rows where S_0 is the next state. And we are constructing this set corresponding to S_0 being the next state and we are capturing here those present states that lead to S_0 as the next state. Notice that S_0 is there as the present state and that is all right that does not change this argument in any way.

If it occurs both as in next state and that present state the our argument is still the same which is that there is a cause for S0 and S1 to be close in terms of encoding irrespective of the fact that it occurs on the next state also. This was an argument that related the 2 present states. So, that is still there it does not matter that one of them is also in the next state. Fine, that is my next state set the construction for S0, I can build the same thing for S1, S2, S3 and so, on. I also annotate here the number of occurrences of each of those states in the present state.

So, 2 occurrences of S0 and 1 occurrence of S1 that is what those numbers are ok. So, I can build this for all my states.

(Refer Slide Time: 46:57)

Fanout: Constructing the Graph

- Node = state
- Edge Weight (a-b)
 - multiply node weights in each OUTPUT_SET
 - if a, b are in same set
 - each pair of transitions has a common cube
 - multiply node weights in each NS_SET
 - if a, b are in same set
 - scale by $N_b / 2$

OUTPUT_SET₂ = {S1 (2), S2 (3), S3 (2)}

NS_SET_{S0} = {S0 (2), S1 (1)}

NS_SET_{S1} = {S0 (1), S1 (1), S2 (1)}

NS_SET_{S2} = {S1 (1), S2 (1), S3 (1)}

NS_SET_{S3} = {S2 (1), S3 (1)}

For $N_b = 2$:

Edge Weight (S2 - S3) = $3 \times 2 \times ((1 \times 1 + 1 \times 1)) \times 2/2 = 8$

(C) P. R. Panda, IIT Delhi, 2017 24

Now, I have to build my graph, I have a captures so, far this much. One was an output set corresponding to every output I would have those sets if there were 3 outputs I would have 3 sets. Then I have these next state sets there would be one corresponding to every state ok. well, every state that matches that pattern of a common next state then I have to annotate the edge weights. It is a fully connected graph if that pattern never occurs then it is fine you just have a 0 there on the edge weight right.

So, here is what we are doing in the process of constructing those edge weights, let us say S2, S3 is the pair that is the edge that we want to build the weight for you look for sets in which S2 and S3 both occur together. All of these are providing some argument for S2 and S3 to be close codes remember. So, because this is then a common next state,

that is the common next state and this is a common output all of them have this property in common that they are setting up an argument for making S2 and S3 close.

Now, of course, there is competition now if you look at S0 and S1 similarly another set of pairs could be chosen this is such an example S0 and S1, this is also such an example and so, on. So, since these pairs occur a number of times we are to prioritize them in some way and that is the quantification that we are trying to build. Notice that S0 and S3 never occur together in any of these sets ah, that is why that number is 0 that are telling us that it is not particularly important that S0 and S3 we close in any way, they can be least in terms of the priorities when we choose the encoding.

So, the way the quantification is done is we multiply these numbers. You consider this a pair at a time, if we multiply these 2 we also multiply these 2 numbers. That is what leads to that 3 times 2 there the one times 1 comes from here, the other one time one that comes from there. This is scaled by N^b by 2 that just corresponds to the number of ones in the encoding for that state. That many times you would have that common cube occurring, but actually I do not know how many ones are there the ones and zeros, I still do not know we are still only right now arguing about higher priority and lower priority right.

The ones and zeros will be assigned later. So, I do not know how many of them. So, I will just multiply this by N^b by 2 assuming that half the bits are 1 and the other half are 0. Wherever you have a 1 you will have an appearance of that common cube. You see that the argument is the same even if you had picked up the zeros because the number of zeros are also same. So, that is the scaling that I have there and I ultimately come up with some number 8 that gives an estimate of how many literals I would save, if these 2 where S2 and S3 were close.

Student: So, the significance of half is not very clear will this undirected graph maybe later it is clear the N^b .

Why half?

Student: The N^b by 2 the scale N^b by 2.

Right, right, why did we multiply this by?

Student: How benefit (Refer Time: 51:03) not exactly.

See what are we trying to capture here.

Student: It is a relative.

So, there is something that is quantified right? What I want to estimate is the number of times that will occur. The number of times that will occur we can look at by maybe we can go back to our common next state argument right. We said that in this case there was the possibility of a common cube of size 3. Why 3? The 3 correspond to what was common between S_0 and S_2 that we are not directly controlling this will be the output of our encoding, but there is a other thing that this is occurring in A, B, C.

But it is not occurring in D and E why is that? A common cube of size 3 is occurring for some of those expressions, but it is not occurring for the other expressions why is it not occurring for the other expressions?

Student: The next state (Refer Time: 52:25).

Because, if I look at the encoding for the next state there is a 0 for D and E that is why I did not have these. Now the question is I want to do 2 things, first is how much is there in common? That is the size of the common cube. Other is how many times is that common cube occurring? Ok so; the scaling that I am doing is with the number of times that, I expect that common cube to occur, that is the reason for the multiplication. Now, this is a tricky question because, I do not know how many there were 3 occurrences here just because of my choice of 11100 if that encoding was different then the number of occurrences would be different.

Now, I do not know the encoding right now at this stage I do not know encoding right encoding is going to come later in a separate step. But, I do need to multiply just to give a little more authenticity be more precise. So, what I am saying is I will multiply that whatever my estimate is by 5 by 2, here in this particular case it should have been 3, but that would be a as a result of the choice, but that choice is not yet there, but this is kind of an estimate of the number of times that common cube is going to occur that is my N b by 2.

Student: what I meant was all the edges are getting multiplied by divided by 2. So, it is a relative change even if you do not do it would my relative priority change?

No, there are 2 components of this quantification one is this, the other is the output that is not being scaled.

(Refer Slide Time: 54:07)

Graph Embedding

- Start from Graph we constructed
 - fanin
 - fanout
- Assign codes to minimise Cost

$$\text{Cost} = \sum_{\text{edges } a \rightarrow b} \text{wt}(a \rightarrow b) \times \text{HD}(a \rightarrow b)$$

HD = Hamming Distance

Encoding 1
 S0: 00
 S1: 01
 S2: 10
 S3: 11
 Cost = $3 \times 1 + 5 \times 1 + 2 \times 1 + 1 \times 1 + 0 \times 2 + 8 \times 2 = 27$

Encoding 2
 S0: 00
 S1: 11
 S2: 01
 S3: 10
 Cost = $3 \times 2 + 5 \times 1 + 2 \times 2 + 1 \times 1 + 0 \times 1 + 8 \times 1 = 24$

NPTEL (C) P. R. Panda, IIT Delhi, 2017 25

Student: That is.

Right so, there is a difference between the 2 you are right that if everything was being multiplied by 2 then why have the 2 yeah I could remove it.

Student: Oh I did not notice sir (Refer Time: 54:22).

So, that is my graph the paper does mention a different algorithm called Fanin. This was essentially taking the common next state argument and quantifying it. There is a different argument which is along the same lines, but I am skipping that called Fanin. That would essentially take care of the other argument of a common present state and similarly build an argument and build an undirected graph.

You get in general different graphs for Fanin and for Fanout. You could pick either of them for the encoding, but the objective at this stage is to come up with the graph. That graphs somehow captures an important property that helps us later in the encoding, it

tells us that if that number is high, if that edge weight is high then it is higher priority, addressing that pair is higher priority in terms of the encoding.

If that number is low then the priority is low.

Student: Sir one more.

Yeah.

Student: here the outputs were fixed 0 and 1, somewhere a decision has happened that my output has to be 0, 1, but typically when we are designing the architecture and things we have flexibility of deciding the output states also when 2 FSM's are going to interact like when I am partitioning the bigger design into smaller design I have a flexibility of changing the output bit definitions also it is up to me.

Yeah in fact, even here in our context of hls there where did the FSM come from this is the output of the hls step right. Do we have a choice with respect to these outputs? What are these outputs? Anyway from the FSM where are they going? They are going into the data path they become what in the data path?

Student: (Refer Time: 56:15) control signal.

Thus control signals particularly the select signals of the multiplexers do we have a choice of?

Student: Yes.

(Refer Time: 56:23) at 1 or 0?

Student: Yes.

What is the implication of a actual choosing 1 or 0?

Student: The registers were.

Yeah it is just reorders the inputs in a different way. Certainly we have the flexibility to reorder the input. So, you could in fact, just like the symbolic states you could have the outputs also be in.

Student: When will have.

Symbolic.

Student: Divided by 2, 4 output set also in the place just to confirm.

Yeah you anyway if once you have a symbolic you could do the similar argument for ah.

Student: I wanted to confirm that (Refer Time: 56:50).

Yeah of course, the FSM synthesis is taking the FSM in a general manner, then that you have the flexibility to encode or not is a different question. And it turns out that in our context as in many other contexts you do have the flexibility, but after all a finite state machine there is a formal definition of what a finite state machine is and this algorithm of course, is targeting that formal definition where explicitly you have ones and zeros and do not care on the inputs and outputs ok.

So, the second part would be what is called a Graph Embedding problem. You start with that graph that you constructed and you want to assign codes to each of those states in a way that the total cost is minimized somehow. What is the cost? We can easily use the hamming distance of the encoded state along with the weights that we have assigned here in this graph to come up with a cost in this way for. So, sum over all the edges that you have in that graph the product of these 2 things, 1 is the weight right if it is 2 then there is the weight and the 2 is multiplied the weight is multiplied by the hamming distance between those 2 states.

You can see that if the weight is high then to minimize the cost it is worth keeping a low hamming distance and if that weight is a small like the weight is 0 then it does not matter you can keep a high hamming distance. So, that would be the cost function that we would like to minimize in such a graph embedding problem. So, we can easily motivate it using 2 random encodings say you select one of these. So, that is a valid encoding 2 bits are being used and of course, they are distinct.

So, that is one valid encoding. The cost can be computed like this there are some overall the edges you get a number like 27 of course, the reason this was chosen in this way is that you have an expensive encoding for that particular edge that had a high the transition that had a high edge weight right. intuitively what should we be doing in terms of this

encoding? Since, since this is this way it is high we should be assigning codes to S1 and S2 that are minimally distant.

But, this encoding is not doing that that is all. So, the the weight is relatively high if you choose a different encoding particularly make the distance between S1 and S2 small, then you go through the same things you will end up with a lower total cost. So, the graph embedding problem is to select encodings in such a way that that cost is minimized.

Student: The weight of H is more than gain factor.

Yes.

Student: So, that gain we will have no control over after we have these (Refer Time: 60:12).

Yeah, we are assuming that this is fixed now the graph is given and once the graph is given then it is actually a graph problem. It represents our synthesis being modeled in this way, but graph embedding indeed is a more general problem just like graph coloring that we were able to translate it into such a known problem and we can solve it using one of the algorithms that is available.

Student: The gain is a function of a input graph that is the state tables that are (Refer Time: 60:43).

Of the state tables.

Student: We do not have control over the encoding.

We do not have a control over the gain well we did control it in the sense that it is the output of our quantification. Where did those numbers come from? We made that calculation right for the Fanout algorithm we said that this is how I will calculate and we put those numbers there.

Student: But.

So.

Student: (Refer Time: 61:10) fixed (Refer Time: 61:11) I mean.

It is fixed, but we chose that particular strategy to quantify. There is nothing sacrosanct, what is fixed is the structure of the FSM that is fixed. Now from there we said that if you have common if you try to extract common next states and so, on then you quantify in this way. In fact, an alternative algorithm is also there for the Fanin which would actually result in a different graph for that same FSM. That is where there is a the possibility to play around a little bit in terms of what you want to prioritize.

Student: That point we are trying to minimize the gain factor between edges.

Yeah.

Student: And ah.

Yeah.

Student: Once we are confirm the axes then the gain switch then here we are trying to fix the (Refer Time: 61:37).

Yeah.

Student: (Refer Time: 61:38).

Assuming that the gains are all somehow nicely captured in that graph now the problem is just to find the encodings.

Student: I mean.

Right.

Student: Distances.

Right.

Student: That will be minimum.

Right.

Student: (Refer Time: 62:12) highest rate.

Right in fact, the way this problem is formulated you can see that it is not clear if you can see that, but there may be many solutions if you can find one solution you can find other solutions too, that have the same cost. Is that clear? why is it? So, clear or is it not clear?

Student: (Refer Time: 62:30) found one solution for the cost.

Right, suppose you find one solution that minimizes the cost you can argue that you can find other solutions for the same cost.

Student: (Refer Time: 62:40) different (Refer Time: 62:41).

Yeah, what is the guarantee that you can find?

Student: (Refer Time: 62:45) 1 0 1 and 1 1 0 again it is a (Refer Time: 62:48) distance in between the.

Right.

Student: Cost.

If all the encodings were inverted you would have the same the argument here is only in terms of the hamming distance not in terms of the absolute code that you are assigning and therefore, the problem will admit multiple solutions ok.

(Refer Slide Time: 63:08)

Solution to Graph Embedding Problem

- Well studied problem
- Practical observation
 - for large FSMs, clusters of nodes with high edge weights exist
- Heuristic – in what order to choose nodes for assignment?
 - identify heavy clusters and assign codes to their states
 - remove these states from consideration X
 - proceed with remaining graph

NPTEL (C) P. R. Panda, IIT Delhi, 2017 26

Why do we encode the problem in this way in terms of the graph embedding problem? This is another one of those well studied problems that you do not have to work out the solution. Difficult it belongs to that class of difficult problem you can see that a large number of choices are there exponentially large number of choices are there.

However, the heuristic that we choose can be related to some property that we expect for these graphs. These are not just general numbers that we expect on those edge weights, but there could be some expectation. Particularly when you have large FSM's the practical observation here is that you expect clusters of a nodes with high edge weight.

All the edge weights are unlikely to be balanced this is just the nature of the kind of FSM's that practical FSM's that we have. You could in theory construct an FSM that gives any graph there, but practical FSM's have this what property? That even if you have a 100 states in your FSM, having 100 states is not impractical you could have hundreds of states. But, they are not all densely connected to each other all the states are not densely connected to each other there it is sparse in terms of the transitions that you find even if the number of nodes is large.

There could be N^2 transitions in the worst case, but you do not find N^2 transitions for a practical designs. So, it is partially populated right that would lead to even that graph also being sparsely populated in terms of nonzero edge weights that you can expect. So, clusters of nodes with high edge weights are expected in that graph that could be used to guide our heuristic in some way. So, this is a heuristic decides in what order to choose the nodes for assignment because if you choose one order there are a number of encoding that are not available for the nodes that come later in that order.

Ah so, somehow you are prioritizing those nodes and our strategy then is you identify the heavy clusters and assign codes to their states of those clusters first. heavy means what a priority is high that is why the cluster became heavy. Then I just remove those states from that graph and I just proceed encoding the remaining graph that is all that I do yes.

Student: It cannot be delete all the 0 weighted edges.

Student: Do not form those (Refer Time: 65:40).

Yeah it is a same thing.

Student: (Refer Time: 65:42) it will reduce.

Yeah not very different it is an implementation thing for efficiency purposes you could remove the (Refer Time: 65:48).

Student: (Refer Time: 65:50) nodes which are densely connected.

Right.

Student: We are optimizing the.

Right, this is only an informal notion. So, this has to be formally identified, what do you mean by cluster? heavy cluster means what? I have to have a little calculation to tell me what is heavy, but this is what is the overall heuristic.

(Refer Slide Time: 66:12)

Embedding Algorithm

1. Select node n with maximum sum of N_b incident edge weight
2. Assign some code to n and minimally distant codes to the N_b adjacent nodes
allows all N_b nodes to be at distance 1 from n
3. Remove node n (and all incident edges) from graph
4. Repeat (Step 1) with smaller graph

Graph details: Node b (pink) has incident edge weights 4, 6, 2. Node a (green) has incident edge weights 3, 2. Node c (orange) has incident edge weights 4, 0. Node d (green) has incident edge weights 3, 2. Node e (green) has incident edge weights 3, 1.

Select b (max sum: $4+6+2$)
 $b - 000$
 $a - 001$ (distance from $b = 1$)
 $d - 010$ (distance from $b = 1$)
 $e - 100$ (distance from $b = 1$)

NPTEL (C) P. R. Panda, LLT Delhi, 2017 27

Here is what I do, that is my graph, select a node with the maximum sum of N_b incident edge weight. What is N_b is 3 means that I use 3 bits to encode this. So, what you are saying you select a node with the maximum sum of 3 incident edge weights. Why 3 we can decide later, but it is like this over here you would select b , that is my prioritization ok. Why b because if you add the 3 heaviest edges that are connected to it, it would be 4 plus 6 plus 2 right that 4 actually is not there. So, so these 3 states these 3 edges.

The fourth one is 0 so, that does not count the other 3 counts. So, that is my heaviest node. So, I select that node N and you assign some code to it does not matter as we said

the absolute value does not matter. So, let us just assign all zeros 0 0 0 is the code for b that is not important, but the what is important is what we are doing next. In the choice of these 3 connected nodes, they are the heaviest ones that are connected to b. Heaviest means in terms of their relationship with b that is heaviest. So, we can satisfy those 3 relationships at least by assigning codes to a, d and e those 3 nodes that are minimally distant from b that much at least we are able to.

As you can see this is not guaranteeing optimality, but is a nice heuristic in the sense that you are selecting. how did you select b? Where probably because among the edges that are there those are the heavy ones that you managed to select. And this informally is the cluster that we are talking about. Now you see why it is N b? Why the heaviest 3 edge weights as opposed to the cluster which could also be the sum of the all the edges that are connected.

Student: That is what.

Why not the sum of all the edges? Why did I say 3 edges?

Student: Because the heaviest weight node is connected to 3 that is why which was (Refer Time: 68:29).

Because the flexibility of giving uni-distant codes is there only for 3 others.

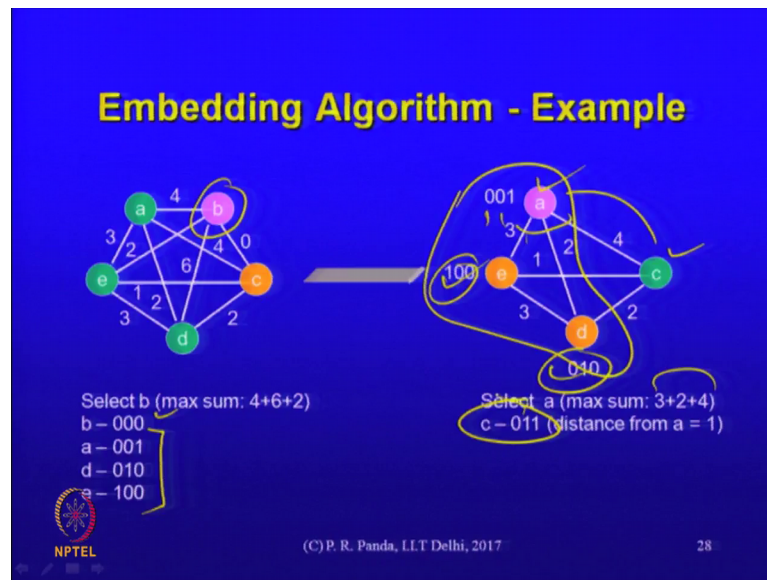
Student: yeah.

Right you do not have that flexibility for the fourth.

So, that is the reason for the heuristic yeah. You could also have chosen the fourth one, but then you would again have to prioritize the fourth one, you would not to be able to give a distance of one you would have to select something else.

So, but this is the reason for the choice. Ok, after that I just remove b from the graph and I just repeat the first step.

(Refer Slide Time: 69:04)



So, simple heuristic start with this, the choice of that node means actually in the process I have done most of the encoding, not only have I chosen the encoding for b, but I have also chosen the encodings for ad and e. But, I remove only b from the graph and continue knowing that for part of the graph the encoding is already done.

Now, for the remaining nodes that are there I will go through the same process which means that a is selected here as my heaviest node because, of the sum of still N b here. So, sum of these 3 as it turns out 2 of it is neighbors are already encoded nothing to do for them and I choose for c, the minimally distant and coding from a that is still available not yet used.

Yeah. So, these 2 are already taken up you cannot use that, but whatever else is available among those you can choose one that is minimal. So, if you want to find other encodings that are at a distance of 1, you could invert this 1 and as it happens that is available even this 1 you could invert. So, such a an encoding can be chosen for c that is essentially this embedding strategy. So, conceptually what we did is in a first stage we looked for patterns and built a graph why go through those 2 stages it simplifies the formalism somehow.

Particularly the second part is totally unrelated to the first one, it is assuming that you did a good job in prioritizing those where computing those edge weights the. So, that the

second part is totally decoupled you could use a simple or complex whatever kind of algorithm you want and those are the 2 conceptual stages.

There is not much contribution in the second stage except that we reuse one of the known solutions, but the contribution as usual in these synthesis problems is in a good modeling of your particular problem into a graph problem so, that we can use some existing solution. So, the quantifying of those edge weights is really what is happening in this algorithm.