Design Verification and Test of Digital VLSI Designs

Prof. Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 3
Logic Optimization and Synthesis
Lecture - 6
Multilevel Implementation

Good morning and welcome to the 6 lecture on module 3, which is on Multilevel Implementation. So, in this in the module 3rd module of the design part of the lecture what we have seen we have seen that given any binary function binary function. So, what you have to do, you have to try the represent it in represented in terms of some minimized forms by minimize form means, we have to represent in the minimum number of product terms and also each having minimum number of literal.

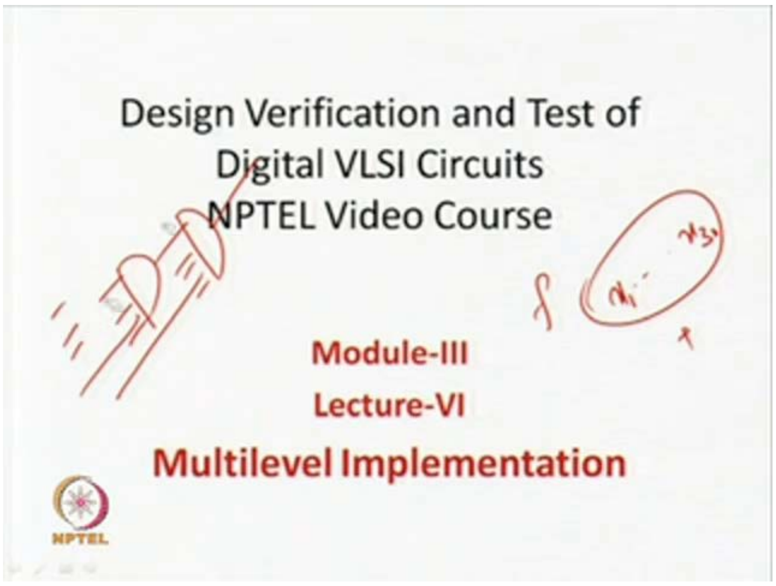
So, this will lead to what is situation called and implementation in terms of minimum number of gates then, we have seen the exact amount of algorithm both heuristics and what you say exact algorithm kind of stuff for 2 level Boolean function minimization. So, whenever we mean by what do you mean by 2 level implementation means, there will be AND gates series of AND gates followed by OR gates or if we the sum of product form or if we the product of some forms then you have sum of the AND gates followed by the OR gates and so forth.

I mean that that means, if you are having this 2 edge of representing representing the functions like sum of product form or product of some form and generally, you are using 2 levels to represent them, first may be the set of OR gates followed by AND gates. So, that is actually have some term followed by the product terms or the other way you should round have you should have you can have some AND gate, that is the product terms followed by the some OR gate is the sum of the those terms.

So, they are can be they these are sum sum of product or product of some terms. But, they are all in 2 level implementation then, we have found out that whether although, this 2 level implementation the algorithm exist like, that is there are the queue go for tabular method that and then you go branch and bound method or if you go for the heuristics method.

So, they can give you fairly well minimize function, but also we have pointed on those lectures that, if you go for 2 implementation, there is some practical problem in the implementation view. So, what it will happens, if you have say for example, if you have function, where is the 1 function.

(Refer Slide Time: 02:14)



Design Verification and Test of Digital VLSI Circuits NPTEL Video Course

Module-III

Lecture-VI

Multilevel Implementation

I mean in a function in f one of the product term may be say x 1 dot dot dot x 30 and compasses there. So, if we actually involve something like a 30 input and gate. So, whether although this is possible in a 2 level implementation it may be a some kind of minimized form, so, but if you that the idea that, you require a 30 input and gate as practical in terms of implementation. So, in in terms of various implementation the it is said that any gate having more then 4 fan outs not desirable.

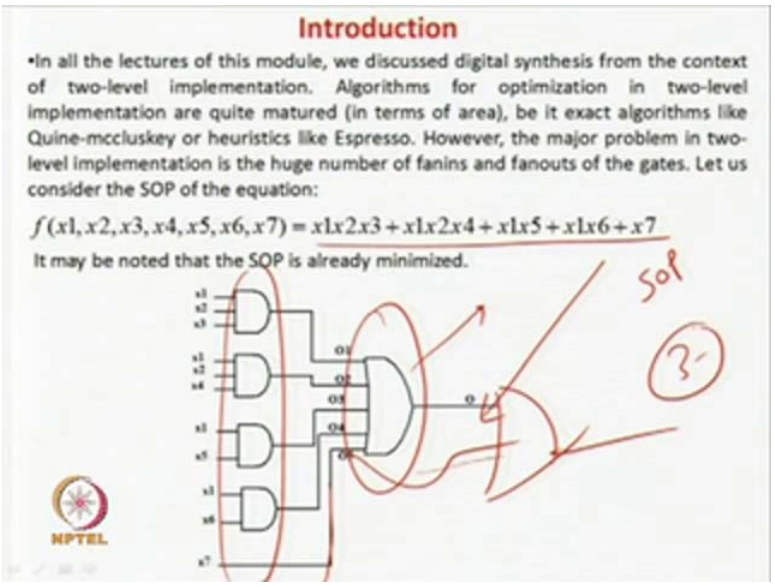
So, what do you require to do so, if you are in 30 level input AND gate. So, you have to break it up into levels of 4. So, 1 for input AND gate, there be other gate with 3 more inputs and then the another this 4 and so for. So, have to completely break it up to multiple levels. So, what is the idea that, if you directly go for 2 level implementation. So, sum of the inputs sum of the gates as high as are very high much, much more then 4.

So, it is found out that will say y, so it will found out that, it is go for such type of implementation where, the number of fan ins are of gates or more then 4, there is out of practical problem in implementation that, it will be slower and that, it will lead to high power and so for. So, what go for that is go for something multilevel implementation. So, in case what happens, we first take a circuit or function, we minimized it in terms of 2 level fundamentals only that is by .

If you are using heuristic branch and bonds what ever fix to you, but as you know that, branch and bond is the highly time complex algorithm so then generally, in the the variance are used. So, the use some algorithm and them minimized for 2 levels implementation, then what you do may find out the some of the gates may have the large number of inputs. So, which is not practical to be implemented then, you have do what we have visit, if the break the 2 level, the 2 level implementation into multilevel implementation and you have keep it in mind a such that.

None of the gate should have more then some amount of pan outs for example, 4 is the very good that, you should not have more then 4 input fan in for any kind of this, because that not lead to that gate becoming the slow, in terms of speed and also it make high power consuming and so for. So, you will see, so let us take a sum of product form this is a product form.

(Refer Slide Time: 04:22)



Introduction

•In all the lectures of this module, we discussed digital synthesis from the context of two-level implementation. Algorithms for optimization in two-level implementation are quite matured (in terms of area), be it exact algorithms like Quine-mccluskey or heuristics like Espresso. However, the major problem in two-level implementation is the huge number of fanins and fanouts of the gates. Let us consider the SOP of the equation:

$f(x1,x2,x3,x4,x5,x6,x7) = \underline{x1x2x3 + x1x2x4 + x1x5 + x1x6 + x7}$

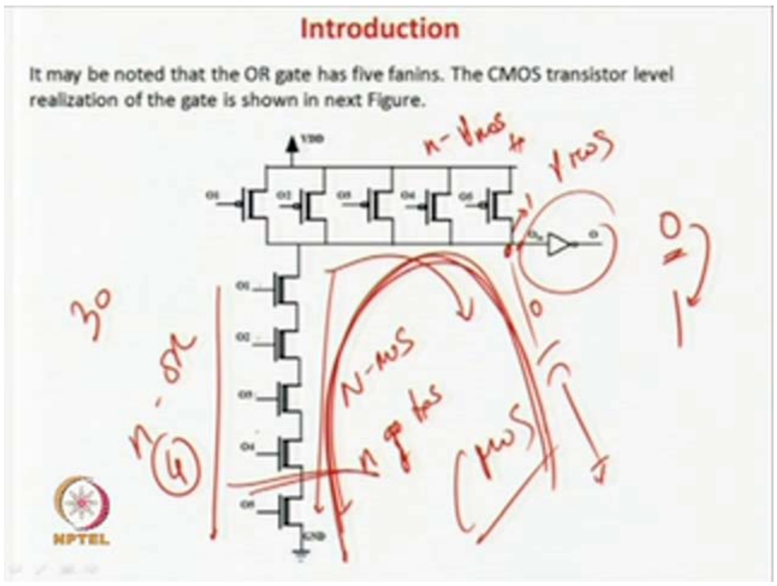
It may be noted that the SOP is already minimized.

So, in this case, if you see, this is already minimize sum of product form, you can take it from the , we find out that the find that, you cannot go for further minimization of this. So, now, if want to implemented some implements the function in terms of 2 level implementation. So, is the sum of product form sum of product form. So, in this case, you have call this product terms here and this is your sum term, because it is odd you can find out the number input of the OR gate 1 2 3 4 and 5.

So, this is the 5 input OR gates, now what are the problem, there is not problem, but implement this is mathematical term or it is minimized 2 level implementation, but the idea is that implement in this or gate with 5 input the big problem. Because, this would any gates in the more the number of these the gate is the slow in operation and power consumption is the higher, so what will do is that, we try break it up into 2 OR gates may be.

So, what do can you do is into have a good implementation, you can have have you can put other OR gate over here and this 5, you can put it over here and that is the idea. So, now, in the 1 2 3 4 will be in faver of and so there the large fan, you put it over here, in the OR gates. So, the same function, we will have represent that, the level 1 2 and just 3 level of the implementation. So, the you have to break it up in the into multiple levels.

(Refer Slide Time: 05:31)



Introduction

It may be noted that the OR gate has five fanins. The CMOS transistor level realization of the gate is shown in next Figure.

So, whenever the idea is that, so whenever I mean, you have NAND gate give fan in some more 4 may be break it up into large number of levels. So, that in the number of fan of all the give as remains 4 or less. So, let us the why dies it happen these actually, C MOS implementation of an OR gate. Now, this CMOS implementation of the 5 input OR gates at 1 2 3 4 5, so this is the OR gates 5 inputs are there, this is CMOS input of the OR gate. So, this is inverter because, a mean if you.

I i mean, if you are not going to details how the gate design, because this is some analogue V L C A or you can find out the any stand that V L S A design book that, how are AND gate OR gate NAND gate, how they are implemented in CMOS most level. So, in CMOS which are seen that NMOS transistor and this is your PMOS and this is inverter, because generally, we have a is a NOR gate followed by inverter, which is make an OR gate. Now, what is you just see what happen, so if it is n input OR gate of kind nothing or an input AND gates whatever.

So, in this chain you have n gates transistors sorry, n transistor in most and this will have parallel in P MOS transistor that, this is the idea, now this is this is. ok. So, whenever the in gate output is be 1 on the another way, now for this this transistors is used to be charged the actually, the capacity over here, this is used to charge capacitor or you want output of this point the 1 in the conversion will be 0.

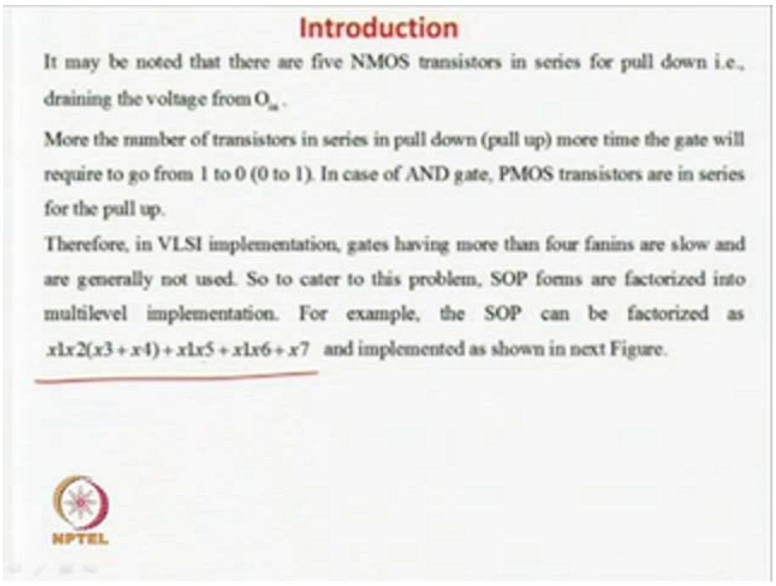
But, that is 1, if you want when you want here to be 0 at the input value here to be is to be 1, so in this case what you have do you have this parallel transistor will charge this capacitor to be so, the answer will be 0.

So, that is you there is 5 parallel transistor there you may be using for charging and when you get the answer it is to be 1 over here, then you have give the value of 0 over here. So, in this case what happen you have to discharge this capacitor to this transistor, now this NMOS transistor, now you see 1 2 3 4 5 more the number of fan out sorry, fan in more the number of NMOS transistor in chain will be there.

So, if we 30 input NOR OR gate number of transistor will be 30 and the time taken for discharge for this path will be very, very long. So, if want that output on the from the 0 to 1, the time taken will be very, very long, because you have to go out the restarting this in most transistor. So, as in the 5 input or gates, so, number of transistor in series is 5. So, that is why we try to limited 4. So, that the discharge path were become very long any obvious if you for more number of transistor for requirement will be very high.

So, the lights, if you any very very high, the fan in gate is very very high at the discharge path of the will be very very high. So, the transition required is going to be very large, this will make the operation of the gate slower.

(Refer Slide Time: 08:05)



Introduction

It may be noted that there are five NMOS transistors in series for pull down i.e., draining the voltage from O_{ss} .

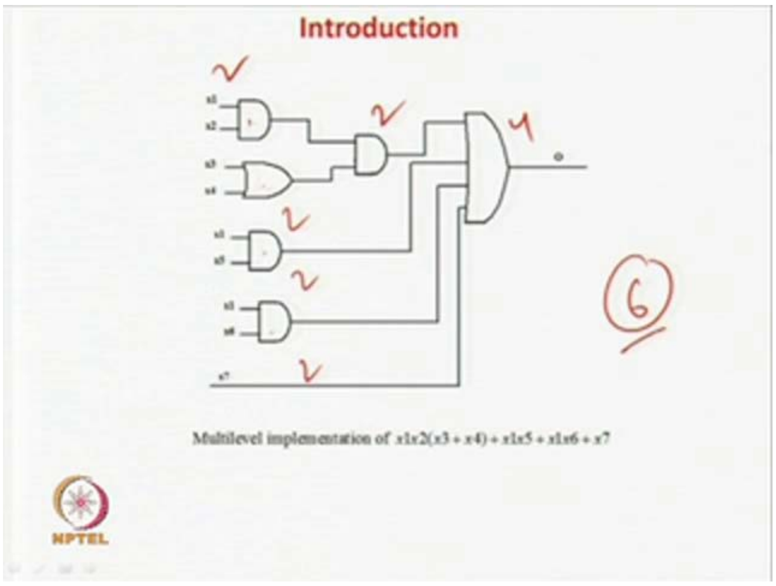
More the number of transistors in series in pull down (pull up) more time the gate will require to go from 1 to 0 (0 to 1). In case of AND gate, PMOS transistors are in series for the pull up.

Therefore, in VLSI implementation, gates having more than four fanins are slow and are generally not used. So to cater to this problem, SOP forms are factorized into multilevel implementation. For example, the SOP can be factorized as $\underline{x1x2(x3+x4)+x1x5+x1x6+x7}$ and implemented as shown in next Figure.

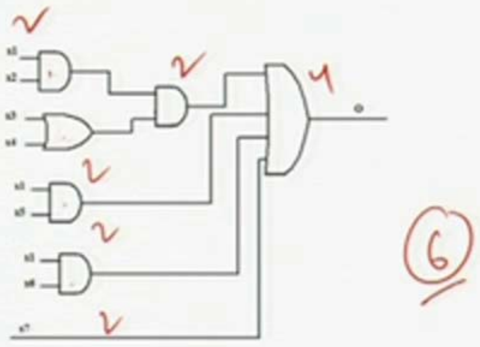
So, that is why you try to leave it our input fan ins to 4. That is what is the standard do, which will taken V L S I design, so whatever I i told you return in this slide. So, therefore, we require a multiple level implementation of this in this function. So, one very standard way of doing it is you just I i mean, very what you have doing, this you have just forget do not add this 5 pin here, in the or gate you just do it over here, but this is very kind of per rule of doing is that every level has 3 level and every gate as 4 inputs.

But, in this lecture, we try to doing the much more standard way. So, that what we can say that all the or that is the multilevel implementation is also very much minimized fashion, so that means, what again the number of product terms or the number of cubes will be minimum as well as the number of be also be minimum. So, that is how we do.

(Refer Slide Time: 08:52)



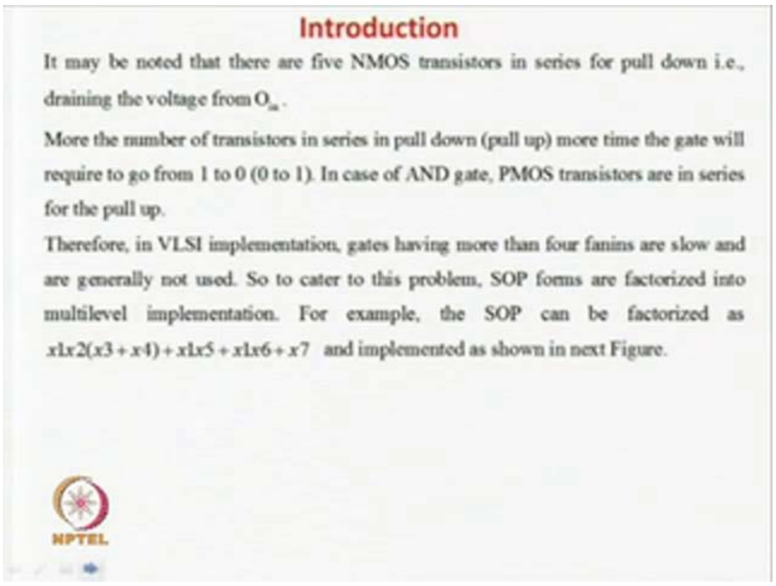
Introduction



Multilevel implementation of $x1x2(x3+x4)+x1x5+x1x6+x7$

So, in this case, if a break it up into and the this OR gate break it up break it up to the into 2, we will found at that I i mean, we will see, in in this case what you have to done. So, we have just taken this into and OR gate and we have just break in up to this. So, you will have doing it.

(Refer Slide Time: 09:05)



Introduction

It may be noted that there are five NMOS transistors in series for pull down i.e., draining the voltage from O_{ut}.

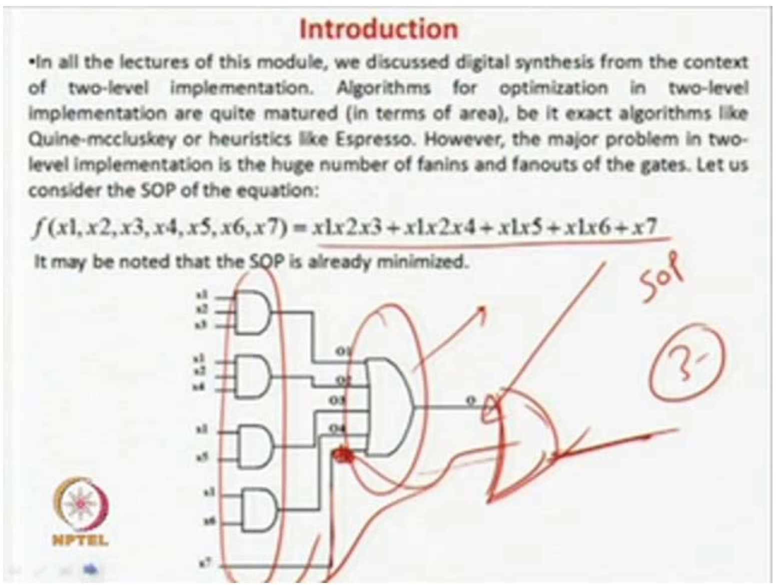
More the number of transistors in series in pull down (pull up) more time the gate will require to go from 1 to 0 (0 to 1). In case of AND gate, PMOS transistors are in series for the pull up.

Therefore, in VLSI implementation, gates having more than four fanins are slow and are generally not used. So to cater to this problem, SOP forms are factorized into multilevel implementation. For example, the SOP can be factorized as $x1x2(x3+x4)+x1x5+x1x6+x7$ and implemented as shown in next Figure.

Now, what you have try to do much more standard way. So, if you do in standard way find out that, the implementation will be much simpler or involving the much number of gate. So, that is the what is the minimization is here, that the minimization will be you have to make it as multilevel format, that that is the sum of the product form, the product of sum, you will take implement in a multiple level. So, it will factorized and also the factorization should be such that the implementation, we minimum now number of gates, that is minimum number of terms, and also the number of of the term should be minimum.

So, there the minimum number that fan ins is the OR gate, same thing like your 2 level minimization at here, that the level should be much more in number.

(Refer Slide Time: 09:44)



Introduction

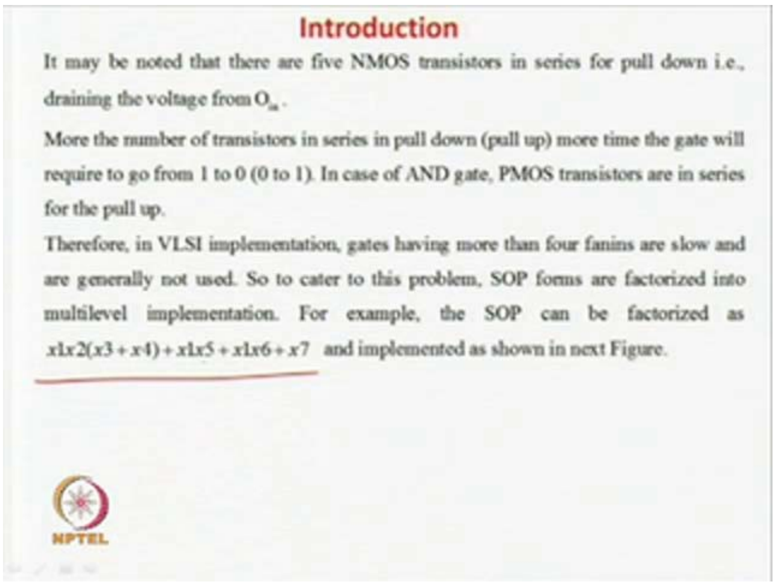
•In all the lectures of this module, we discussed digital synthesis from the context of two-level implementation. Algorithms for optimization in two-level implementation are quite matured (in terms of area), be it exact algorithms like Quine-mccluskey or heuristics like Espresso. However, the major problem in two-level implementation is the huge number of fanins and fanouts of the gates. Let us consider the SOP of the equation:

$f(x1,x2,x3,x4,x5,x6,x7) = x1x2x3 + x1x2x4 + x1x5 + x1x6 + x7$

It may be noted that the SOP is already minimized.

So, this is this a way of doing that you find out the gate have in more then 5 inputs or the 4 the 4 input the you break it up to 2 levels and you keep on doing this at adhere way of doing at will be shortly see. So, for the same function, what is same function this one.

(Refer Slide Time: 09:58)



Introduction

It may be noted that there are five NMOS transistors in series for pull down i.e., draining the voltage from O_{in}.

More the number of transistors in series in pull down (pull up) more time the gate will require to go from 1 to 0 (0 to 1). In case of AND gate, PMOS transistors are in series for the pull up.

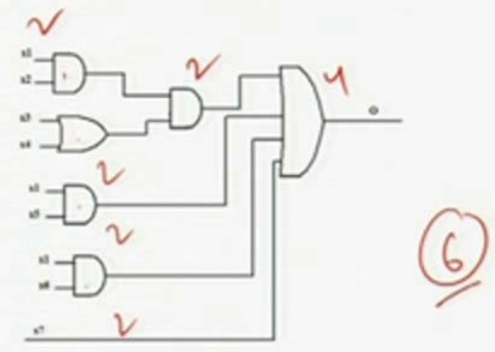
Therefore, in VLSI implementation, gates having more than four fanins are slow and are generally not used. So to cater to this problem, SOP forms are factorized into multilevel implementation. For example, the SOP can be factorized as $x1x2(x3+x4)+x1x5+x1x6+x7$ and implemented as shown in next Figure.

NPTEL

What we these people have done different way. So, in case this is the implementation, same thing they have taken x 1 x 2, they have common part of they can need out x 3 plus x 2 x 1 x 5 and x 1.

(Refer Slide Time: 10:06)

Introduction



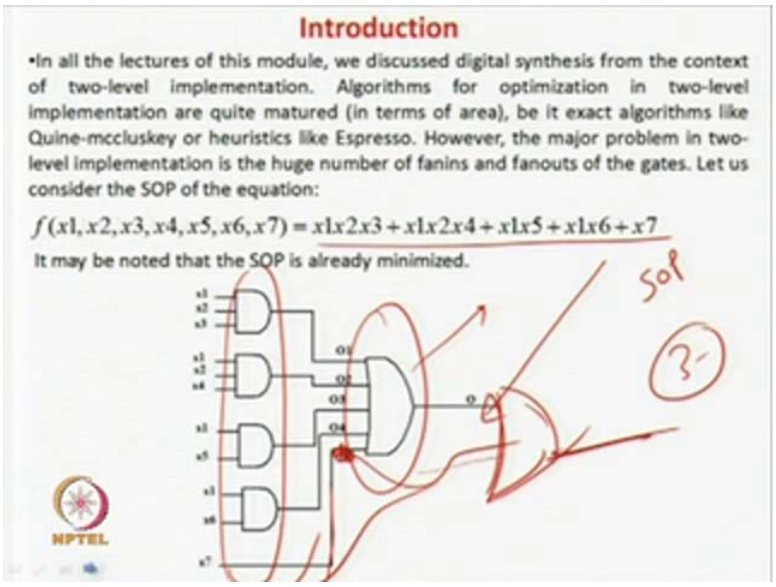
Multilevel implementation of $x1x2(x3+x4)+x1x5+x1x6+x7$

NPTEL

And this will be implemented in this fashion. So, in this case you see, how many number gates are river 1 2 3 4 5 6, 6 gates are there. And in this case also, if you check is 1 2 3 4 5, 6 6 gates are there and 1 very important thing is so, this this implementation is adhere 1, which a cut this OR gate fed it here. So, in this case is 1 2 3 4 6 gates are there and in this implementation also, in this fashion that is again you have broken up into the multiple levels, but you have use the factorization over here.

So, in this case, you can see 1 2 3 4 5 6 gate are required or the number of fan ins of the gates, you check it is 2 here, it is 2 here, it is 2 here, it is 2 here and this 2 here and 4 over here.

(Refer Slide Time: 10:45)



Introduction

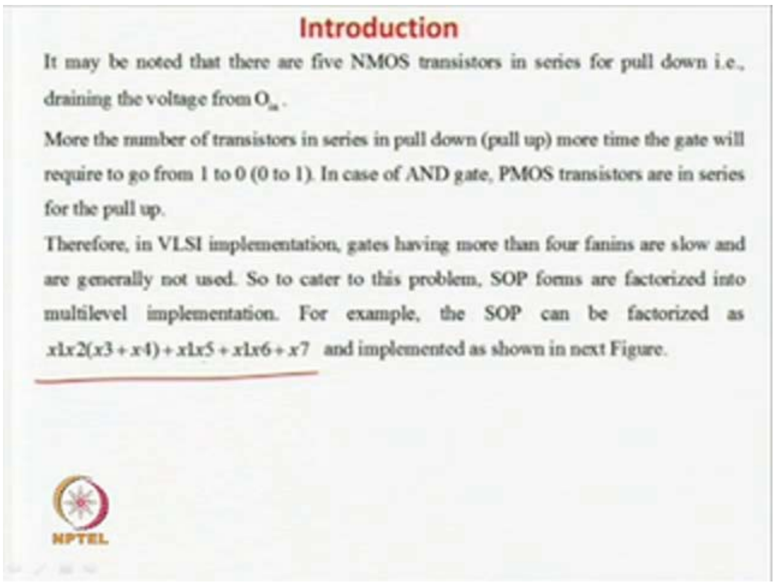
•In all the lectures of this module, we discussed digital synthesis from the context of two-level implementation. Algorithms for optimization in two-level implementation are quite matured (in terms of area), be it exact algorithms like Quine-mccluskey or heuristics like Espresso. However, the major problem in two-level implementation is the huge number of fanins and fanouts of the gates. Let us consider the SOP of the equation:

$f(x1,x2,x3,x4,x5,x6,x7) = x1x2x3 + x1x2x4 + x1x5 + x1x6 + x7$

It may be noted that the SOP is already minimized.

In this case, if you check check the number of input is 3 over here, 3 over here, in this case this is 4 over here. So, the number of fan ins is that is the fan ins of the gates is higher in this case. So, that why if you just take the adhere manner, that wherever you find out the gate with more then fan outs, you break it.

(Refer Slide Time: 10:59)



Introduction

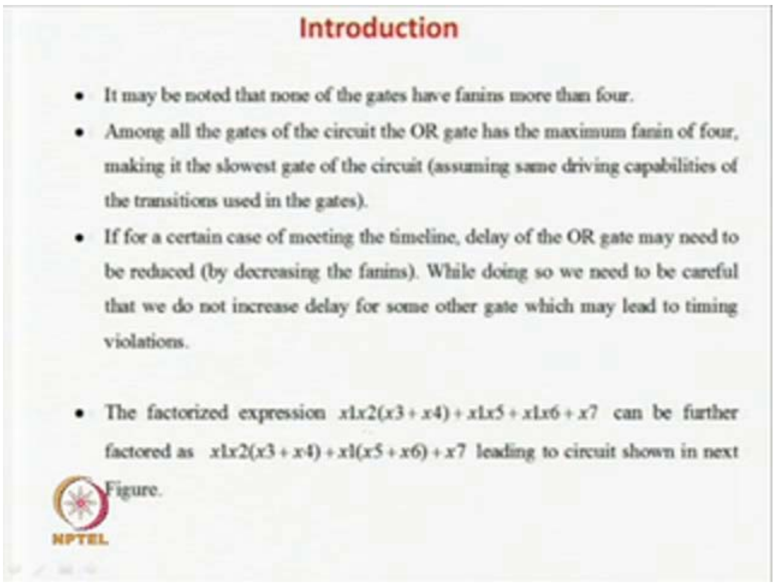
It may be noted that there are five NMOS transistors in series for pull down i.e., draining the voltage from O_{ia}.

More the number of transistors in series in pull down (pull up) more time the gate will require to go from 1 to 0 (0 to 1). In case of AND gate, PMOS transistors are in series for the pull up.

Therefore, in VLSI implementation, gates having more than four fanins are slow and are generally not used. So to cater to this problem, SOP forms are factorized into multilevel implementation. For example, the SOP can be factorized as $x1x2(x3+x4)+x1x5+x1x6+x7$ and implemented as shown in next Figure.

Whether the much more area or or the gates of more number of fan outs or terms , if you go the, this type of factorized implementation (Refer Slide Time: 11:08). Where you have a less number of fan ins, in this example. So, in this example both have 6 gates, but in the more practice situation, we will find out the number of gates is also be lower in this case. So, in this expression what do you have found that, none of the gates have expression a fan ins more then 4 that, was the desire thing. So, everywhere you have to keep it in mind. So, no where you should have a fan in of more then 4 gates, so if it there then you have keep out factorize it.

(Refer Slide Time: 11:30)



Introduction

- It may be noted that none of the gates have fanins more than four.
- Among all the gates of the circuit the OR gate has the maximum fanin of four, making it the slowest gate of the circuit (assuming same driving capabilities of the transitions used in the gates).
- If for a certain case of meeting the timeline, delay of the OR gate may need to be reduced (by decreasing the fanins). While doing so we need to be careful that we do not increase delay for some other gate which may lead to timing violations.
- The factorized expression $x1x2(x3+x4)+x1x5+x1x6+x7$ can be further factored as $x1x2(x3+x4)+x1(x5+x6)+x7$ leading to circuit shown in next Figure.

Then this factorization expression also be further factorized in to this. Then it is just you see (Refer Slide Time: 11:36). This is a beauty of this is 1 level factorization then again, you can find out these common term over here. So, you can again factorize, this in this term reading to the circuit here.

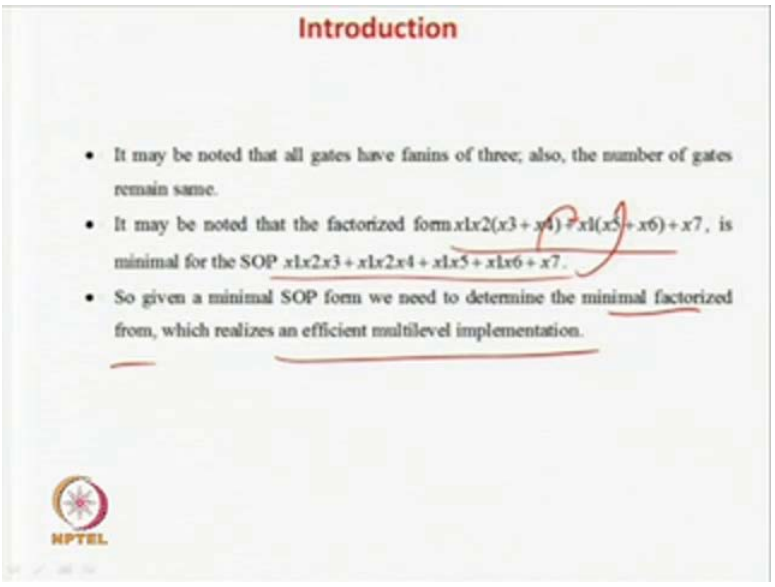
So, in this case, you see 1 2 3 4 5 6 gates are involved in there, but also you can see that, number of fan ins are decrease may much over here, this is 2 2 2 2 1 2 2 2 2 1. So, in this case 1 2 3 4 5 6 7 8 9, so 9 9 input in the this case 1 2 3 4 5 6 7 8. So, in this case you have see a 2 plus 2 4 5 6 7 8. So, there is actually, 6 sorry, 8 fan ins the input gates where is wherever it was actually, 1 2 3 4 5 6 7 8 9.

So, even by taking the common factor on this one and factorization so, again we will found out the number of input fan ins become a 8. That is what is I am saying that, if you keep on factorize the then I i mean, there are going then the adhac way of doing keep of factorizing is as and find out that number of fan ins in the inputs are decreasing also none of gates have more then a further number of fan ins like 4, in the this is .

In other, but more formal what you try to achieve, we are trying to achieve the minimal representation of this function (Refer Slide Time: 12:48).This is this your function to the try the represent it in a minimal multi level multiple level format. So, when you say minimize represent the minimize representation will be such. So, that it cannot be re factored in it. None of the terms here can factorized further so that means, what is factorize in the best way. So, it will implies as between minimum number of minimum number of minimum number of terms implying the minimum number of gates and minimum number of fan ins. So, that is what in case of 2 level implementation of same thing, we are trying to reduce the terms and also we are trying to reduce the numbers of in there.

But, in this case as its multi level implements, we have to reduce the terms and also reduce the number fan literals, but at a same time you have to do factorization, because you are going first multilevel implementation and the factorization should be such that no more terms can be re factored in it. That is the minimum number of that is almost, you have bottom levels. So, that the number of we have achieve a you can achieve a minimal representation in terms of gates, now, that is the idea.

(Refer Slide Time: 13:47)

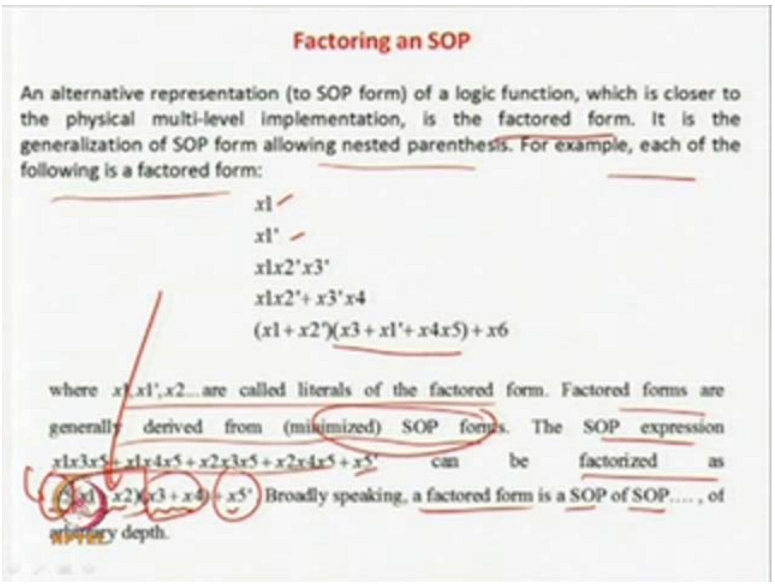


Introduction

- It may be noted that all gates have fanins of three; also, the number of gates remain same.
- It may be noted that the factorized form $x1x2(x3+x4)+x1(x5+x6)+x7$, is minimal for the SOP $x1x2x3+x1x2x4+x1x5+x1x6+x7$.
- So given a minimal SOP form we need to determine the minimal factorized from, which realizes an efficient multilevel implementation.

So, this is this is the actually, the gate we are factorized term for this 1. So, this is minimal representation, because no terms factorize any more. So, this is the minimal factorize from, which is realize in the multi level implementation. So, that is the target so, we will a given us sum of product form, in this way you have to go for a maximum factorization form, that is what is the minimization in multi level implementation. So, this is this was what you have discussed by what you say as the small example.

(Refer Slide Time: 14:11)



Factoring an SOP

An alternative representation (to SOP form) of a logic function, which is closer to the physical multi-level implementation, is the factored form. It is the generalization of SOP form allowing nested parenthesis. For example, each of the following is a factored form:

$x1$

$x1'$

$x1x2'x3'$

$x1x2'+x3'x4$

$(x1+x2')(x3+x1'+x4x5)+x6$

where $x1, x1', x2$... are called literals of the factored form. Factored forms are generally derived from (minimized) SOP forms. The SOP expression $x1x3x5+x1x4x5+x2x3x5+x2x4x5+x5'$ can be factorized as $((x1+x2)(x3+x4)+x5')$ Broadly speaking, a factored form is a SOP of SOP...., of arbitrary depth.

Now, we go to the for algorithm where we try to the show how can we do factorization. So, that is a factorizing a sum of product form, it is also be done for a u s form, but mainly in this lecture, we are concentrating on the sum of product form. So, before going that go for some standard definition an all that, we go for the exact algorithm. So, an alteration representation of the S O P for from of a logic function, we just closer to the physical multilevel implementation is a factored form.

So, if it the P O S or occurs in the 2 level, if you will going for a factorised term it is more level to multi level implementation. It is the generalization of S O P form, which is allowing nested parenthesis for example, each of the following is a factorised form factor it 1 literal 1 literal 3 literals and this is also 1 2 3 4 to cubes.

You can say and that 4 literals, this case it is 1 2 3 3 term and 1 2 1 2, this case 1 2 3 4, 4 terms and 4 literals, so, for you can do this. So, this 1 are the literals of the factor form factored form as generated as drieved from S O P terms minimazation S O P term. So, that is why when, if somebody ask you a question that, if I i do it multilevel implementation then why you will go for 2 level implementation the idea is that, you cannot directly jump into multilevel implementation 2 level implementation is still the vary a date that Boolean function minimization.

So, what you do first you given S O P form you have to go for a 2 level minimization and from the 2 level minimization, you can form, you can go for multi-level implementation and you have to minimize that is given a Boolean function may . So, you have to first obtain the minimize S O P form that is the minimize 2 level implemented form and form the 2 level implementation minimize form, you can again factorize to the maximal level. So, that it becomes a minimal multi-level implementation format, so that is why 2 level implementation till date as a very great role in in minimization of circuit implementation.

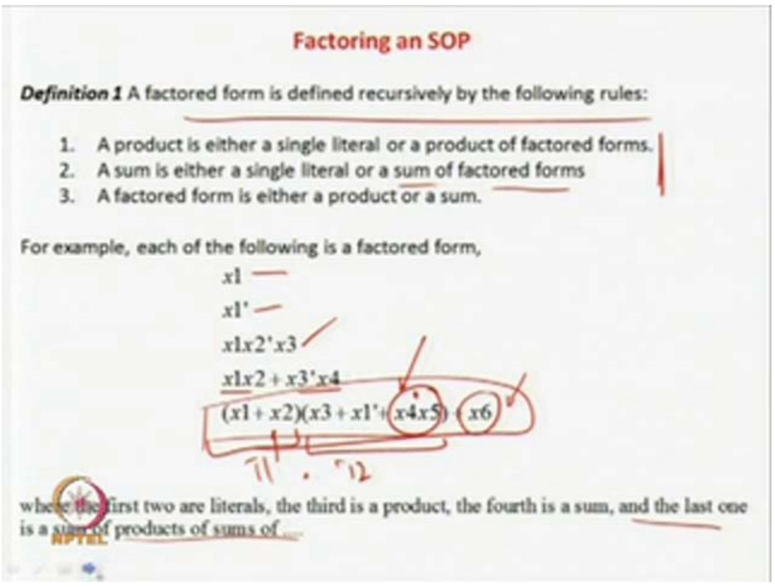
So, that is why factored form has derived from minimized S O P forms. So, minimization of a forms can only be desired, if you all going to 2 level minimization algorithm, if you go, if you are using 2 level minimization algorithm then all you can get minimization S O P forms. So, the S O P expression this one can be factored as also this is another example they have saying x 1 x 3 x 5 and this one. So, you can break it up as x 5 as you can take it as common between this x 5 prime is alone, this x 5 is common everywhere.

So, you can take this then you can find x 1 plus x 2 and x 3 plus x 4, if you can factored the you can get it. So, basically what happens this represent the factorized in this one. So, basically a factor form is sum of product of sum of product of sum of product dot dot dot. So, you can see this is again the sum of product form like x 1 plus x 2 is the sum of product form.

In this case you can also, this is a sum of product form again, you can see that, this is nothing, but a sum of product form and the whole thing is also, you have say sum of product form. If you can think this is 1 term, this is 1 term and this is this is 1 term then is nothing actually, sum of product form. You do not consider the inside the bracket just consider this 1 as 1 term, this 1 as 1 term, this 1 is 1 term and this is single term and this is the single term.

So, this dot, this dot, this this is a sum of product form, but now inside in the element again, you can break it up into sum of product form and this you can in a nested now, that is the factor form.

(Refer Slide Time: 17:24)



Factoring an SOP

Definition 1 A factored form is defined recursively by the following rules:

1. A product is either a single literal or a product of factored forms.
2. A sum is either a single literal or a sum of factored forms
3. A factored form is either a product or a sum.

For example, each of the following is a factored form,

$$x1$$
$$x1'$$
$$x1x2'x3$$
$$x1x2 + x3'x4$$
$$(x1 + x2)(x3 + x1' + x4x5) + x6$$

where the first two are literals, the third is a product, the fourth is a sum, and the last one is a sum of products of sums of ...

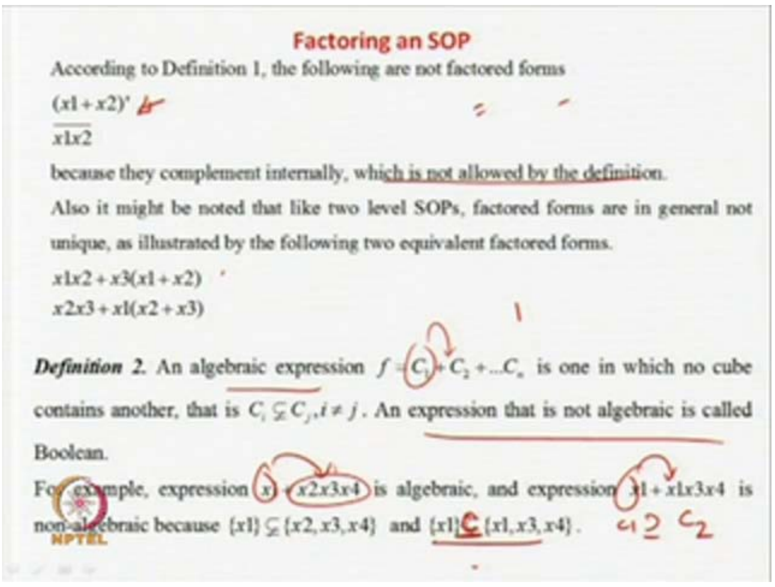
So, now factored form is can be defined in the following way a product is either a single literal or a product of factored forms is very obeys a sum is either a single literal a sums of factored forms a factor from a factored form is either a product or a sum. So, these in 3 rules, you can find out what is the factored form. So, very example this this is a single literal the factored form, this again a single literal factored form this 1 is either single literal or sum of product of this, you can see this is the sum of factored forms. So, in this case it is a sum of sum factored form.

So, this is also a factored term and again this is actually, sum of product sum of product sum of product. So, this also nothing, but a factor form, this is what is the a definition that is a recursively, you can define a factored form in this way. So, that why I i told you this is 1 product tem, this 1 tem and this is 1. So, you are this is the product of 2 terms and this is the third terms, you can say that the this is a sum of product form. So, this is a single literal and this is say term 1, this is term 2, this is the product in the some.

So, in this some of product form, now inside in this blocks again is the again the term 1 term 2, this is again product form. So, this is a again a sum of product form and you keep all on way you. So, as this first literal out the third is a product. Product is the sum you can and the last one is a sum of product of sum of product of sum of product.

So, these are nested definition as you given in here as a nested definition of a fact factored S O P. So, that is why you have to conceive when you looking up into the top level. So, you have to the term and this term 2 and this is term 3 the sum of product forms, now again if you go inside the 1 sum of product forms what this is a term in a single literal here. If you go to again term, 2 this again a sum of product forms, these 2 a single literals and this is a product term. So, these how you can the factored form.

(Refer Slide Time: 19:09)



Factoring an SOP

According to Definition 1, the following are not factored forms

$(x1+x2)'$

$\overline{x1x2}$

because they complement internally, which is not allowed by the definition.

Also it might be noted that like two level SOPs, factored forms are in general not unique, as illustrated by the following two equivalent factored forms.

$x1x2+x3(x1+x2)$

$x2x3+x1(x2+x3)$

Definition 2. An algebraic expression $f = C_1 + C_2 + \ldots C_n$ is one in which no cube contains another, that is $C_i \subsetneq C_j, i \neq j$. An expression that is not algebraic is called Boolean.

For example, expression $x1 + x2x3x4$ is algebraic, and expression $x1 + x1x3x4$ is non-algebraic because $\{x1\} \subsetneq \{x2, x3, x4\}$ and $\{x1\} \subset \{x1, x3, x4\}$. $c_1 \supseteq c_2$

So, which are not the factored form that is also very important. So, according the definition this is not a factored form, because if this you can open it, up this can be opened as because, this is x 1 plus x 2 whole complements.

So, this can you cannot represent something like this follow these definition, this C 1 can be represented as a x 1 bar dot x 2 bar , you can find then this will be come a factored form, similarly this one also is not a factor form. Because, you can see that is x 1 dot x 2 whole bar, this will be another thing, but x 1 and x 2, this one is this one plus this one. So, also this one will your something like the. So, this mean you can open in it happen to your de Morgan’s slow. So, can find the doubt then this become your factored form. So, this is if you have some expression like this.

So, this is not S O P form then this will not your factored form. So, because their complementary entirely, which is not allowed by the definition also it might be noted that like 2 level S O P factor forms, in general are not unique as illustrated by the following 2 equivalent factored form. So, we know that is for a single function that can be a multiple way of implementation in 2 levels as you already seen. So, in same way factors forms, they are not unique, because uses these 2 or find out their equivalent.

So, this one is nothing, but x 1 x 2 this one will open it a x 3 x 1 plus x 3 x 2, now you just open it up also you will found out that, it is x 2 x 3 plus x 1 x 2 plus x 1 x 3. So, you can find out the this equivalent. So, in other words, so just like us 2 level implementation form. So, S O P or P O S form they are not unique for a given function. So, multiple level implementation is also not unique. Now lets go to the next definition. So, it say that an algebraic expression, this is f of c 1 plus this is nothing, but c cubes, this is one.

So, an algebraic expression is 1 where no cubes are contain the other that is now one of this cubic, totally emended another there is an algebraic expression an expression, that is not algebraic is called Boolean. So, in case of Boolean 1 may be included in the other for example, this 1 is algebra because this one is not in this another, this one is another term. And expression this one is known as algebraic, because this whole trem, you can say is inside this as it true.

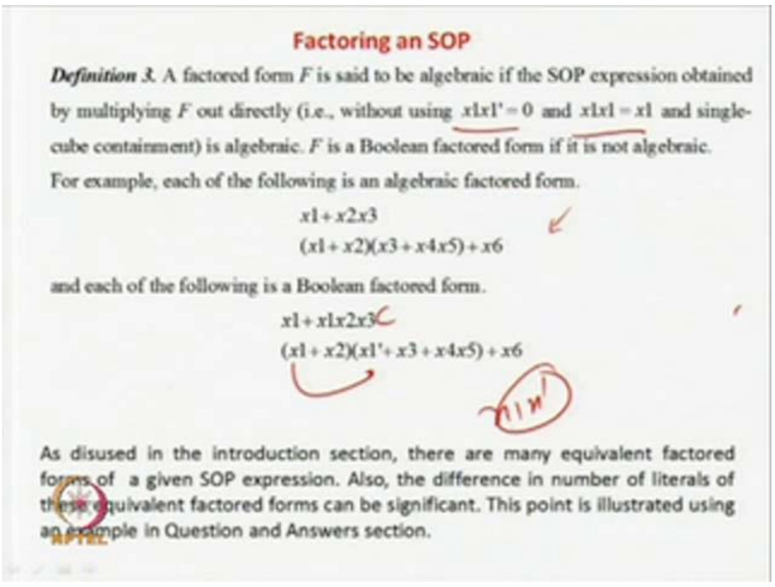
So, in this we know algebraic this one not algebraic, this one is algebraic, because x 1 is not in this one and in this case sorry, it will be something like this you can say. So, this is the case sorry, the type over here. So, this is not algebraic, because x 1 is not included here and this one is this one is Boolean not an algebraic first one is sorry, these expression is algebraic, because x 1 is not a part of this one in another words.

So, this this term this cube is not the inside, this that was the algebraic and the expression this one is a non-algebraic expression this one is a non-algebraic y, because the whole c in this implying that, this one is case of this what is written that should not be there is type of my say sorry, for that. In this case this is this one belong to this one. So, x 1 does not belong to this this implies that C 2. Is that is that is C 1 sorry, this was means that means, none of the literal from C 1 is in C 2 and vice versa.

So that means, what is C 1 is this will not hold in C 2 that means, what what a mean over here is that as x 1 is not as this literal, present in C 1 is not in C 1 C 2 or x 4 and that will mean that C 1 will not belong to C 2 and so, for. And either C 2 will be belong sub set of this one. But, now you can see in this is neither, this is also true, this this is also be true over this, because, x 1 is not in this form form, in this, but x 1 is what if you see at the 2 expression like expression this one.

X 1 plus x 1 plus x 1 plus 3 and 4, this one in this case this not algebraic, this is Boolean because, this ah literally a parts of this this one implies that, if you consider the C 2 C 1. So, in this case this will be the case. That C 2 will be inside C 1. So, in this case you will remain a Boolean.

(Refer Slide Time: 23:25)



Factoring an SOP

Definition 3. A factored form F is said to be algebraic if the SOP expression obtained by multiplying F out directly (i.e., without using $\underline{x1x1' = 0}$ and $\underline{x1x1 = x1}$ and single-cube containment) is algebraic. F is a Boolean factored form if it is not algebraic.

For example, each of the following is an algebraic factored form.

$$x1 + x2x3$$
$$(x1 + x2)(x3 + x4x5) + x6$$

and each of the following is a Boolean factored form.

$$x1 + x1x2x3$$
$$(x1 + x2)(x1' + x3 + x4x5) + x6$$

As disused in the introduction section, there are many equivalent factored forms of a given SOP expression. Also, the difference in number of literals of these equivalent factored forms can be significant. This point is illustrated using an example in Question and Answers section.

So, now there are another definition a factor is F is said to be algebraic, if the S O P expression obtained by multiplying F out directly. That we will see that does mean is algebraic and is Boolean, if is not algebraic, that is if you given a factor form something like this now, if you open it up. If you multiply this that is this thing should implement and this thing should not be there, that is instead of minimization you should eliminate that is if you have x 1 plus x 1 make it 0, it is eliminate of x 1 x 1 keep does a x 1 do not may keep as x 1 dot x 1.

And single cube to containment of those things of that is all this type of minimization you should do and if then then if a becomes a algebraic expression. Then it will be the an algebraic factored form become a Boolean factor form for example, which have the following is a algebraic factored form like this is x 1 x 2 x 3. So, if in if I i opened up, the already open. So, x 1 does not belong to x 2 or x 3. So, this will become an algebraic format, now if you get this 1.

So, if you can open it up this will be 1, it will be x 1, if you just open it up this is x 1 x 3 plus x 1 x 4 x 5 and then if you open it x 2 x 3 plus you can have into x 3 x 2 x 4 x 5 plus x 3. So, you can easily find out that none of the term has totally, inside other. So, that is what we have saying an algebra, this is saying which is no cube contains an another. So, in this you can very easily find out the that, I i mean in the none of the cube contain each other.

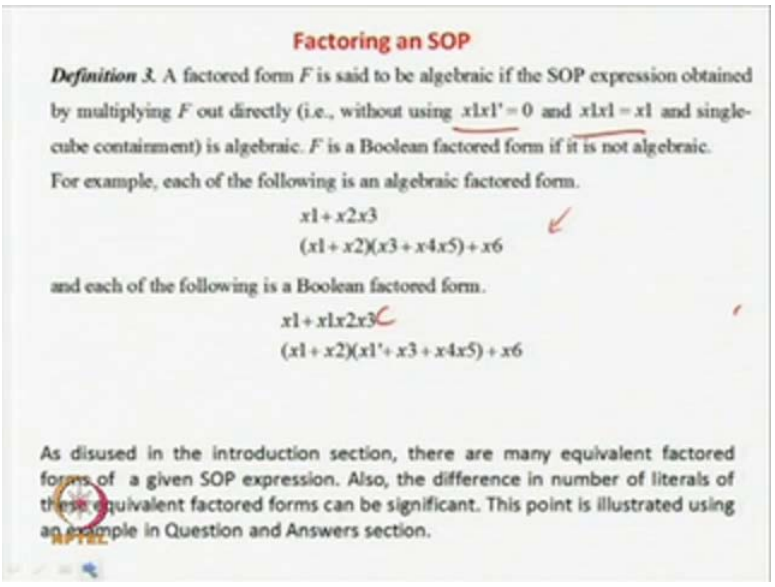
so in this case this is obviously the and we have there is not term like this one. So, that we not think about it like for example, we will not have the, if you have something like this one, we have to eliminate this term. So, as there nothing is there. So, we can easily say that these 2 are algebraic format algebraic factored form, because none of the cube contains another. The following the Boolean factored form, because these are the obviously, if you look at this already, we have told that mean, this if you can say this is C 1, this is say C 2 C 1 will comprise C 2.

So, in this case this is not. Similarly, you can easily find out that, in this case also, if you open it a. So, you will find out that this is a now there all algebraic factored form, because first way you will find out the x 1 x 1 prime. So, this will actually kill you. So, you can easily find out, I i mean, if you just go ahead open this expression. So, you can easily find out that, this factored from some this will provide a definition to in this find out the this is the Boolean factor from an non algebraic factor form.

So, now, is that, so in a factor form. So, deference the number of literal can be significant. So, what what is you can what is the idea, it is said that as discussing the introduction section, there can many factor that was given S O P significance also the differs in the number of the literals of this equivalent forms can be significance, that is already given in a example, that is you can see, that in this case the input is 8 (Refer Slide Time: 26:29).

You have input in the same thing, if you do into this input input fan ins is the 9, if you are not going for a factor form then you very, very 3 plus 6 8 9 10 11. So, that is for a same function that can be different factored forms and the number of the inputs can be different.

(Refer Slide Time: 26:46)



Factoring an SOP

Definition 3. A factored form F is said to be algebraic if the SOP expression obtained by multiplying F out directly (i.e., without using $\underline{x1x1' = 0}$ and $\overline{x1x1 = x1}$ and single-cube containment) is algebraic. F is a Boolean factored form if it is not algebraic.

For example, each of the following is an algebraic factored form.

$$x1 + x2x3$$
$$(x1 + x2)(x3 + x4x5) + x6$$

and each of the following is a Boolean factored form.

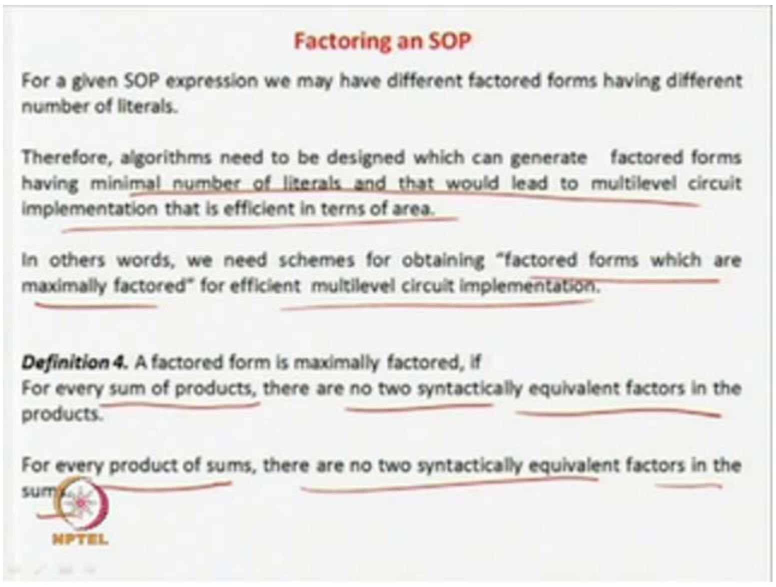
$$x1 + x1x2x3$$
$$(x1 + x2)(x1' + x3 + x4x5) + x6$$

As disused in the introduction section, there are many equivalent factored forms of a given SOP expression. Also, the difference in number of literals of these equivalent factored forms can be significant. This point is illustrated using an example in Question and Answers section.

So, that is why we have also , this has a another example of the question answer and that is why that, we have go for such a factored form where, the number of input fan ins are minimum as well as the number of terms also should be minimum. So, that then only, we can say it is a minimally sorry, maximumally factored form and that will equal the number of implementation.

So, slowly slowly towards that so, we have slowly going towards first, you have seen definition of a Boolean factored form what is algebraic factored form, first you have also seen factored form. So, we have seen what is the Boolean from an algebra form factored form that why you are using, we try to find the algorithms, which can give you what is the best or the maximally factored form, which is the best for implementation slowly will go.

(Refer Slide Time: 27:26)



Factoring an SOP

For a given SOP expression we may have different factored forms having different number of literals.

Therefore, algorithms need to be designed which can generate factored forms having minimal number of literals and that would lead to multilevel circuit implementation that is efficient in terns of area.

In others words, we need schemes for obtaining "factored forms which are maximally factored" for efficient multilevel circuit implementation.

Definition 4. A factored form is maximally factored, if
For every sum of products, there are no two syntactically equivalent factors in the products.

For every product of sums, there are no two syntactically equivalent factors in the sum.

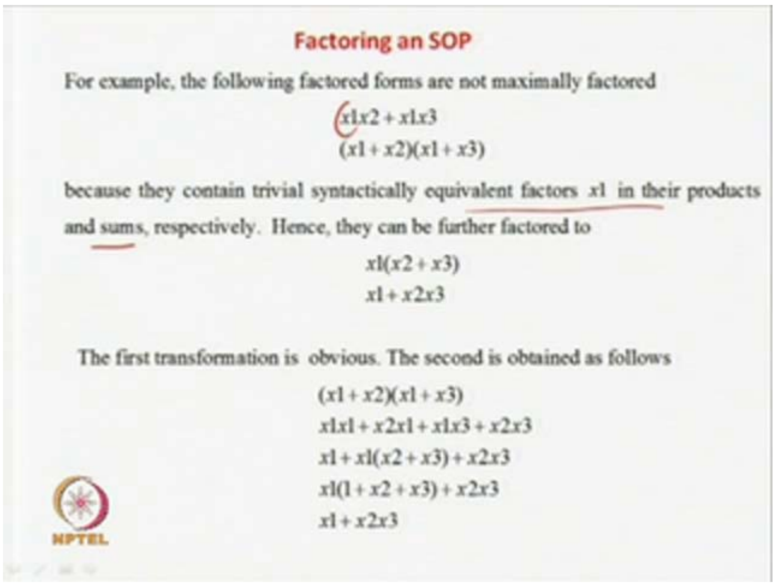
But, you have to keep in mind just like simple S O P sum of product 2 level implementation a product of sum 2 level implementation or minimization that can be there can be different 2 level implementation. Similarly, multi-level implementation also there can be different factored forms give into different type of implementation for the same function. So, therefore, we require algorithm to generate that can generated factored having minimal number of literals that was lead to minimal implementation that minimal level implementation.

So, that is efficient in terms of area in other, we needs sachems for factored from, which are maximally factored for efficient multilevel circuit implementation. So, with algorithms starting now, which can given S O P minimize, so, S O P form, which can go for the maximum factored form maximally factor way maximal factored, they will the minimize the S O P forms and will factorize, it maximally so that the number of literals and terms in them are the minimum is the very less or will be will be minimum as possible for the implementation area in efficient.

So, now let us take the definition what is by a maximum factored form. So, in other way in a is same as language what is the maximally factored of given any some of product minimum sum of product form, it will consider as maximum factor, if none of the terms can be again factored. That is in one way doing it but, this is very laymen of this language saying it, that is once is the. So, must factored no more terms can ne prefecture in it.

So, we let us now, we see how can be define a more form, factor from his maximum the factor, in every sum of product for every sum of product in that factored from there are no 2 syntactically equivalent factor in the products for every product of sums, there are no 2 syntactical equivalent factor in the sums. So, now, will I i mean rather then just going towards the technical does not mean illustrated by example. So, that the we will come back to this definition.

(Refer Slide Time: 29:17)



Factoring an SOP

For example, the following factored forms are not maximally factored

$$x1x2 + x1x3$$
$$(x1 + x2)(x1 + x3)$$

because they contain trivial syntactically equivalent factors $x1$ in their products and sums, respectively. Hence, they can be further factored to

$$x1(x2 + x3)$$
$$x1 + x2x3$$

The first transformation is obvious. The second is obtained as follows

$$(x1 + x2)(x1 + x3)$$
$$x1x1 + x2x1 + x1x3 + x2x3$$
$$x1 + x1(x2 + x3) + x2x3$$
$$x1(1 + x2 + x3) + x2x3$$
$$x1 + x2x3$$

For example, these 2 are not not maximally factored, now why because to the very trivial, because they contains con contain trivial syntactically, equivalent factored x 1, in their in their productions, in their sum respectively, now they are the sums and they are the products. So, you have a syntactically equivalent term over here. So, that is being said every sum of products, there is not to syntactical equivalent factor in the products for every product of sum that is no 2 syntactically, factors in the sums that is what we have the mean say.

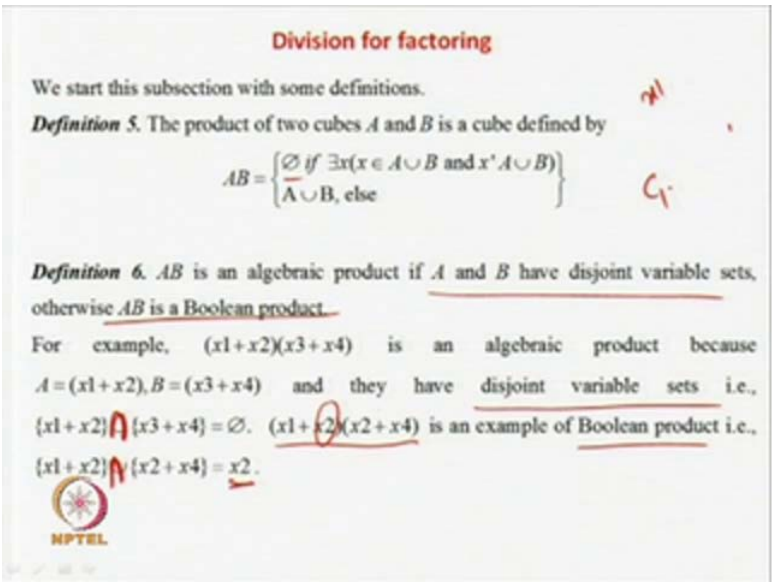
So, in this products so, you can say there are 2 syntactically, 2 equivalent factors similarly, in the sums of also . So, you can say that a sum or a I i mean S O P form is maximally factored, if you do not find any kind of syntactically, equivalent terms of factors in the sum of sum term. So, obviously you can be factored very simple way you can factor this one as x 1, you can taken as common and you will get x 2 plus x 3.

So, this is what we have got this this one, this term this term can be factored into this one. So, mean first is obese and the second one easily find out, you can just open this up then that, you can find out this the most factored implementation. So, the first one product terms at you just take the common 1 and the sum term. So, if you go out go out, this step you will find out.

Now this is the minimally facts maximally factored form, because you can see that there is this is 1 product term, you can consider one product term. There is no any common initially the x 1 as the common syntactic factored syntactical equivalent factored of x 1, but there is no fact, such a factor over here and similarly, in this case of there is non-factor over here. So, you can tell this term, maximally as the factored now obviously, they are going to if you represent them.

So, they will represented in what you can say minimum number of gates will be represent in the minimum number of levels as well as they will representing the minimum number of fan ins that is minimum number, if gates with minimum number of inputs, now these what is it case.

(Refer Slide Time: 31:15)



Division for factoring

We start this subsection with some definitions.

***Definition** 5.* The product of two cubes A and B is a cube defined by

$$AB = \begin{Bmatrix} \varnothing \text{ if } \exists x(x \in A \cup B \text{ and } x' A \cup B) \\ A \cup B, \text{ else} \end{Bmatrix}$$

***Definition** 6.* AB is an algebraic product if A and B have disjoint variable sets, otherwise AB is a Boolean product.

For example, $(x1+x2)(x3+x4)$ is an algebraic product because $A=(x1+x2), B=(x3+x4)$ and they have disjoint variable sets i.e., $\{x1+x2\} \cap \{x3+x4\} = \varnothing$. $(x1+x2)(x2+x4)$ is an example of Boolean product i.e., $\{x1+x2\} \cap \{x2+x4\} = x2$.

So now, more definition, so we say that the product of 2 cubes A and B is a cube, so that the 2 cubes C 1 and C 2 say a product of 2 cube say A and B. So, in C 1 and C 2 are A and B in this case. So, if take 2 cubes, because the standard, we are using C 1 and C 2 4 representing cubes. So, simply if there are 2 cubes C 1 and C 2, in this represent they are terminate A and B is to multiply them.

So, A a dot B that is 1 dot C to is 5, if there exist an x belong A union B A prime union x union B that mean, but actually, if mean you say that, it is a x 1 dot x 2 then this one is say x 1 prime sorry, and this 1, you will say C 2 is say something x 1 prime x 2 sum . So, it is a there an x, which belongs to which is a terms belongs into C 1 union C 2 literal belong into that is A union B and also belongs to. So, if you make a product of this the x 1 dot x 1 prime. So, it will doing into 5, otherwise it will be simple area that is the very obviously.

So, A B as an algebraic product, if A and B has disjoint variable sets, otherwise A and B is A Boolean product. So, obvious if A and B have no terms in common so, if you make a product among them so, it will be a disjoint variables. So, it will become a algebraic product, otherwise A and B is the it will be a Boolean product, because every sum term will be common and obviously the 1 term will include other.

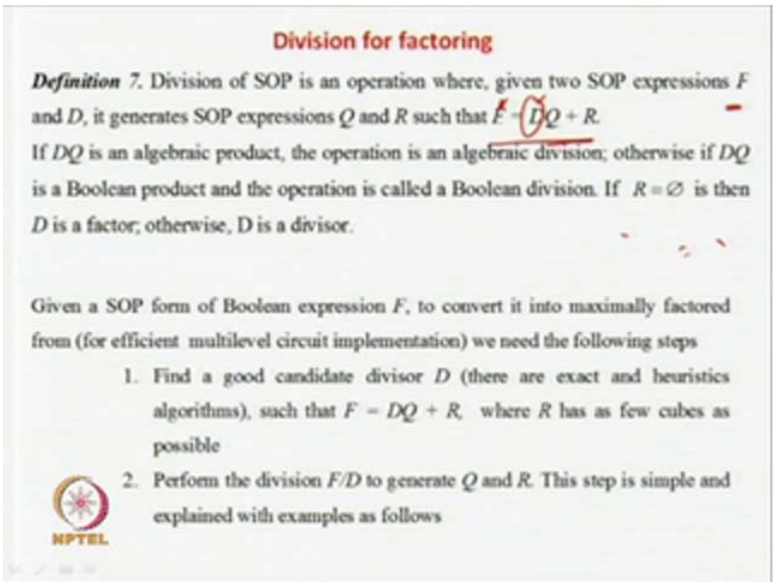
So, you have seen this example. So, this one is a algebraic product, because x 1 plus x 2 be do not have any common literal and there is not common any on variable. So, if you have. So, that is what is I i sorry, this is an intersection. So, this is into 5 and now this example of a Boolean product why because, ah sorry, this is all be intersection, we also the talking about this term.

So, this is all intersection I i am very sorry, because now if we in this case. So, x 1 x 2 x 1 prime x 2 prime. So, at x xi is I i mean, say in this case x 1 is complements. So, if you make your products of it. So, x 1 x 1 prime cancel out and C 1 will be 5 what now if you have x 1 x 2 and in this case, that is x 1 x 3, now if you make product, you will found out that it will be the common part will be only 1, also you can have a thing like this.

So, this is about the idea, you can in this case uses these intersection can has 5. So, this will be a Boolean as well as algebraic factorization, now in this case, this is x 1 x 2 and this one, actually this one is case it is Boolean product, because x 1 and x 2 is to intersection, this will the actually x 2, because there is the common factor out of A.

And so, you are actually these are problem, this is the you have a common syntactic variables. So, you have a Boolean product so, this was the definition.

(Refer Slide Time: 34:09)


Division for factoring

***Definition** 7.* Division of SOP is an operation where, given two SOP expressions F and D, it generates SOP expressions Q and R such that $F = DQ + R$.

If DQ is an algebraic product, the operation is an algebraic division; otherwise if DQ is a Boolean product and the operation is called a Boolean division. If $R = \varnothing$ is then D is a factor; otherwise, D is a divisor.

Given a SOP form of Boolean expression F, to convert it into maximally factored from (for efficient multilevel circuit implementation) we need the following steps

1. Find a good candidate divisor D (there are exact and heuristics algorithms), such that $F = DQ + R$, where R has as few cubes as possible
2. Perform the division F/D to generate Q and R. This step is simple and explained with examples as follows

Now, we are going for the we are now actually, come from using those definition, we are coming to the place, now we wil try to factorize a given this thing. So, now function so, what do you mean by factoriation. So, say if you will go to higher I -i mean junior is school day, if you say that the factorize 15. So, we can say that 15 is equal to 5 into 3 or if you that factorize 20, you say 10 into 2 or some 5 into 4.

So, a factorizing 20 that means, given is the say high school fundamental. So, the same thing is actually, made by factorization, we may also be expression to the expression f. So, this is one S O P form that, you mean function, which have factorize. So, another D is there take 4 the time mean D is division. So, you can find out that a given a name, you want to factorize it. So, some D is given to you that is the divisor. So, you get a quaint also you gets a reminder.

So, but I mean this is very previous simples like a 15 is equals to S, consider a high school days, now some divisor is given to you say D is equal to 3 then question will be in this case will be to 5 and reminder in this case equal to 0. But, if 1 say the divison is 4 then then, what you say then the question will 4 is the the 12 and 13 14 15 reminder, that is equal to do. So, same thing is actually, also holing for this functions. So, what we have doing. So, in this case and division of S O P is operation is made to be S O P, this thing is to generated this and this is your case.

So, if which is the function is the factorize some divisor is given. So, how the divisor is given is the most typical problem, you have say the how is the given a division, you can find the quaint and the reminder. So, what will you have doing what you have try what are the basic idea. So, it will given a function, which we have factorize, now sum way the D has to be given. So, this is very important how the it will selected.

So, in as we seen that the most problem, in finding out the maximally factored multiple level implementation finding of the D id the most difficult problem, if somebody gives you a good division for the time somebody gives you a very good division then what you do. Now, you divide F by D and you get a question and a reminder, now you can try to see that can D b the factor. So, best idea it I i will give you that, you a very good D. So, you can not the D factor much. So, you have the factor Q and R.

So, now, again you have to take you to factorize Q and factorize R again, you have the good division and you have to keep on doing it finally, we have find out that all the terms have factorize and you did not have to go for anymore factorization the maximally factored. So, now, you can understand that selection of the D is very important then all stages. So, in the last we seen expression.

So, the D is an algebraic product and the operation is algebraic, otherwise in the products, this product D Q is algebraic then the division is algebraic like, otherwise this is Boolean product and the operation is called Boolean division. So, way if R is 5 then D is factor, otherwise D is the divisor. So, that is very well known if, so of D can vey well term and the the reminder will will be 0.

So, in this case obese the D is the factor, otherwise these A divisor and the question. So, now an obese in D Q is algebraic product then division is also algebraic, if the product is Boolean product the Boolean is the division, so that is division. So, we already know that what is algebraic product, there is no only syntactic common terms in D and Q and there is common term in the between D and Q then it become a Boolean stuff.

So, as I -i told you, first to take some good D then you do this then again, if you find that Q and R can make it factorize then again, you find out the D prime for D and the D double primes A a for R, you keep on doing it.

So, every time we out the very, we would, we which can doing. So, if you take a very quality of the then then the factorization levels will be very, very slower as we will see much more numbers of steps will be required to factorize it. And finally, you have make a long time to make a good factored form and also, you will find that you may not have receive the most optimal factor factorized , you got an improperly, so that is why selection of these is very very important factor and I -i think in this lecture will not going to elaboration.

How you will find out the very good, because a very well, we lot of work to find out how can you find out a very good D. So, because the algorithm is standing on the seat of how you find a good D. So, how can you find out a good D then factorization, this is also stay forward I -i mean for a division algorithm, which we can follow, which we will see selection of a D is good D is very important.

So, now given S O P Boolean expression is to converted into a maximum factored form, we need the following steps, now we are just that somebody has given you a very good D then how we have this factorization. Find the good divisor D exact and there there are some exact algorithm as well as algorithms actually, is not algorithm, but when you are just giving you very formal terms, there there algorithms and, you say this will algorithm here there will be exact algorithm.

There is no in then a what for the I i mean just for delivery sense and just for I i mean, for talking may not going to that formal computer science, this is more less and wire less iPods.

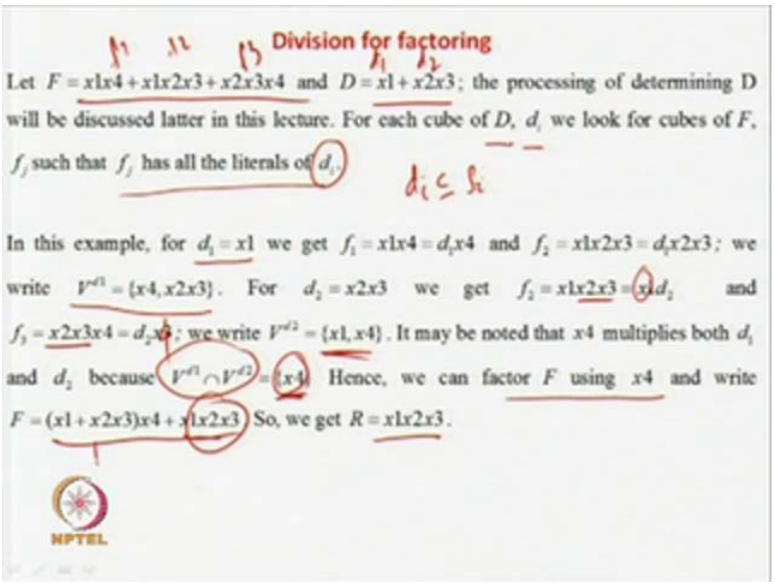
So, I mean, sometimes, we the slide abuse of location I mean, as use of language, we have saying that I mean, exact algorithm and heuristics algorithms, we just have to know that I -i mean, for exact algorithm for this predefined steps and there no choice I -i mean, this may give you optimal solution. But, in case of heuristics, we sometimes go about randomly searching solution it and sometime may not give the most optimum solution.

But, beside solution of language I -i mean for the cede course I -i mean, if you can use this term like exact algorithms and heuristics algorithms. So, there is good anther example of algorithm heuristics to find out good D in this lecture, we give you a vey brief that how it can be done. So, 1 that is there, so you have to go for this and then if you said a very good D then what going to have R is going to have as few cubes as of there that is that, you factorize that a 15 something like this.

So, you now let us take the prime number say 13 factorized such a way. So, that the reminder is less as possible, you may you can say 6 into 2. To the divisor and the plus reminder is 1, you have then if I -i say that, you take 5 the divisor 5, 5 as a divisor. So, it will be 5 into 2 and the remainder will be 3.

So, in this case, if you take a very good divisor, so R will have as a literal as possible. Now, perform the division F by the generate to this may be simple as see how is the division can be done.

(Refer Slide Time: 40:20)



Division for factoring

Let $F = x1x4 + x1x2x3 + x2x3x4$ and $D = x1 + x2x3$; the processing of determining D will be discussed latter in this lecture. For each cube of D, d_i we look for cubes of F, f_j such that f_j has all the literals of d_i.

$d_i \subseteq f_i$

In this example, for $d_1 = x1$ we get $f_1 = x1x4 = d_1x4$ and $f_2 = x1x2x3 = d_1x2x3$; we write $V^{d1} = \{x4, x2x3\}$. For $d_2 = x2x3$ we get $f_2 = x1x2x3 = x1d_2$ and $f_3 = x2x3x4 = d_2x4$; we write $V^{d2} = \{x1, x4\}$. It may be noted that $x4$ multiplies both d_1 and d_2 because $V^{d1} \cap V^{d2} = \{x4\}$. Hence, we can factor F using $x4$ and write $F = (x1 + x2x3)x4 + x1x2x3$. So, we get $R = x1x2x3$.

So, now you will see that also, we will see show you how the division can be done and at same time, we also show you that, if you that taking a very good divisor in a number of times to got to the most factors for by this will factored will be less compare, if you are taking a bad quality division.

So, this is you will say a function, which you have to minimal a factor maxi maximally, factor and this is the divisor x 1 plus x 2 dot x 3. Now, the points determining d will be shown latter, that is the how you can find the good d, we see later just give you the idea here, then for each cube of d now you will see, how it divide d, which cube of d x 1 and x 2 x 3, we look cubes for f such that d has all the literals of f is all the literals of d has.

So, what you will do actually, for d 1 is equal to d 1 let us take the first term. Then we see that f 1 x 1 dot x 4. So, you can represent as d 1 not x 4 right. So, this one, we are repressive by x 1, now x 2 also has x 2 has also x 1, this is f 2. So, this is f, this is f 1, this is f 2, this is f 3. So, what you have try to do for each cube in d, that you have this is d 1 and this is d 2.

For each cube of d 2 look for cubes of f f f j such that, f j has all the literals d i that means, d i actually, becomes a subset equal to f i. So, if you see in this case I -i mean you can factorize f i by d i. So, in this case f i, x 1 is common. So, you can write next d 1 x 4, f is f dot is x 2 dot x 3. So, x 1 is basically d i d 1. So, you can write d 1 d i. So, write the d 1 as this one, that is one is the term you are using. So, d are the d 1 is the cube of the divisor using using d 1 an we x 4 because, first 1 make you x 4.

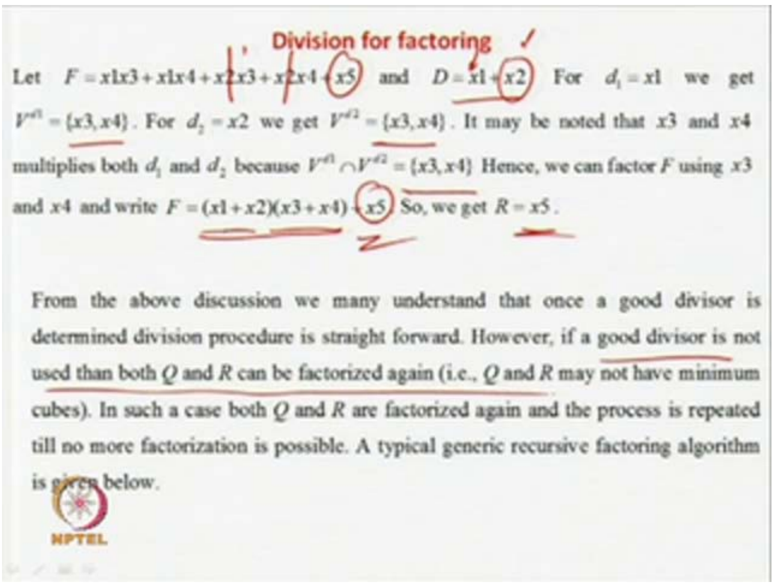
And the second these 2 and the second term x 1 is common, so x 2 and x 3. So, now, So, this is the last term you see f 3 and this f 3 is x 2 dot x3 dot x 4 and this x 1 is no common here. So, you you just cannot do anything with it, now here the next term d 2 is x 2 x dot x 3. So, we get you see, x 1 x 4 in f 1, you cannot doing anything. So, if on you cannot do anything f 2 use a 3 x 2 x 3 and a this one is x 2 x 3.

So, can usually represent x 1 d 2 because d 2 is this one. So, this is the case and also see that, x 2 dot x 3 4. So, in this case d 2 dot x 3. So, we will do be nothing, but x 1 and x 4, because, in this case x 1 and in the final case into a is d 3 x 3 x 4 also, this is what sorry, this is 4. D 2 is nothing, but x 2 dot x 3 so this x 2 dot x 3. So, this is x 4. So, this d 2 is x 3 dot x 4. So, is small type is d 2 not x 3 and this d 2 x 3.

So, in this case we are having x 1 and x 2. So, it may be noted that, x 4 multiplies d 4 and d 1 in d 2. So, actually then you take a intersection of this. So, you get x 4. So, you will find out that x 4, basically multiplies both the d 1 and d 2. So, we can use factored this one x 4 is the common factor for the whole thing. So, you can write in it this way. So, you will write x 4 a you just factorize to it x 4 is equal to x 1 to dot x 3 x 1 x 2 x 3 will reminder, so how you get x 4.

So, what do you have basically, done you have taken the first cube of d 1 and find out the common terms. There is only factored the terms in n, which are, which were possible then I -i have taken x 2 dot x 3. Similarly I i have done the same thing for f 1 f 2 dot f 3 then, I found out that, which is the common factor. So, in this I -i have found out that x 4 is literal, which is common. So, if we taking out x 4, we find out this expression, so this one is becoming is your reminder this one x 1 x 2, this one this was is your divisor and this one is your quotient.

(Refer Slide Time: 43:59)



Division for factoring

Let $F = x1x3 + x1x4 + x2x3 + x2x4 + x5$ and $D = x1 + x2$ For $d_1 = x1$ we get $V^{d1} = \{x3, x4\}$. For $d_2 = x2$ we get $V^{d2} = \{x3, x4\}$. It may be noted that $x3$ and $x4$ multiplies both d_1 and d_2 because $V^{d1} \cap V^{d2} = \{x3, x4\}$ Hence, we can factor F using $x3$ and $x4$ and write $F = (x1 + x2)(x3 + x4) + x5$ So, we get $R = x5$.

From the above discussion we many understand that once a good divisor is determined division procedure is straight forward. However, if a good divisor is not used than both Q and R can be factorized again (i.e., Q and R may not have minimum cubes). In such a case both Q and R are factorized again and the process is repeated till no more factorization is possible. A typical generic recursive factoring algorithm is given below.

NPTEL

So, this is the simple way I -i can actually go about factorizing a term or dividing a term. So, if another example same lyrics and same f and now d 1 divisor is change. So, this is the x 1 and x 2. So, for d 1 equal to x 1, we get x 3 and x 4 right, because if we a in this case of you can cut this x 1, you get x 3 and this x 1, if you cut you will get x 4 over here. So, these 3 terms there is no x 1 x 1 v d 1 is x 2 and x 4, similarly if you consider the this x 2 term. So, in this case mean next will not get anything.

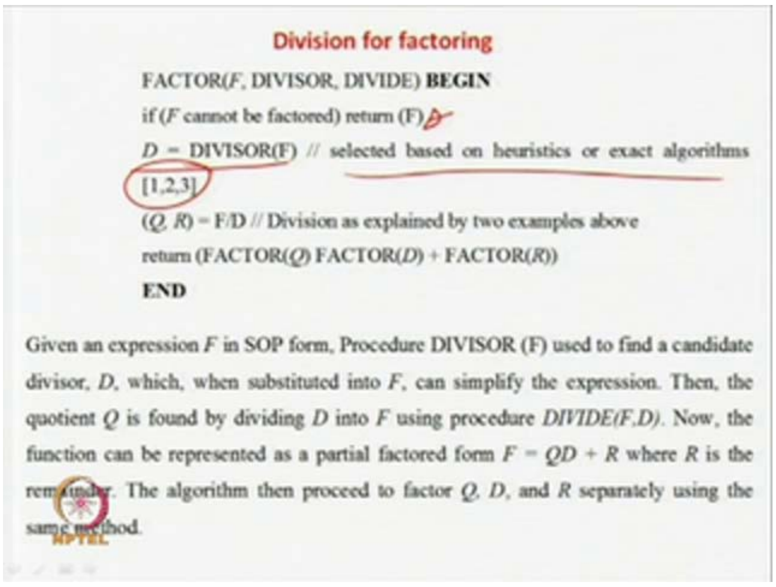
So, in this case, if you cut x 2, you will get x 3, in this case, if you cut x 3 x 4, you will get x 1. So, we to x 3 and 4 x 5 will be no over. So, in this case if we I -i v d 1 and v d 2 try to find out the common term. So, we will find that x 3 and x 4 both are common. So, in this case you can have a factor like x 3 and x 4 is the factor, because common thing is both over here and and you find out the x 1 x will be there and this is the case. So, reminder in this case is x 5.

So, from the ever that was the good division is determine, the division processes is there is state forward, you take a term from here, find out what are the common, where you can actually have a factored form where you can factorize then you make v d 1 take then that term then repeat the same term v d 2 and you keep on doing. In finally, we out what is the common factor about, this what were were the common factor will be that can use to factorize, this term and you can here, that the element.

But, if a good divisor is not used than both R and Q was factorized over here. So, you can see both the example. So, I mean in this case, if you just have a look at it. So, it is x 1 x 2 dot x 3, this one and are is going to this one. So, you can find out the neither this cube. So, this this was divisor this one is the Q and this one is R, neither Q or R here, similarly neither this was D this is Q neither R. So, neither D neither Q NOR R can be factorize, it gain neither in this case. So, in this case you can say that, this D and this D is divisor were very good divisor.

But, if you are not going to get a very good divisor, then what may happen you may be other require to factorize Q and R. So, in this case these not a good divisor then Q and R can be factorize again, they may not be minimum Q. So, you have to do again that in such case Q and R are to factorize again and the process is repeated till no factorization can be done a typical algorithm is given below.

(Refer Slide Time: 46:21)



Division for factoring

```
FACTOR(F, DIVISOR, DIVIDE) BEGIN
if (F cannot be factored) return (F)
D = DIVISOR(F) // selected based on heuristics or exact algorithms
[1,2,3]
(Q, R) = F/D // Division as explained by two examples above
return (FACTOR(Q) FACTOR(D) + FACTOR(R))
END
```

Given an expression F in SOP form, Procedure DIVISOR (F) used to find a candidate divisor, D, which, when substituted into F, can simplify the expression. Then, the quotient Q is found by dividing D into F using procedure *DIVIDE(F,D)*. Now, the function can be represented as a partial factored form $F = QD + R$ where R is the remainder. The algorithm then proceed to factor Q, D, and R separately using the same method.

So, very simple you take a divisor so, these what is the thing, if k is not be factored, you return F that the minimally, of already have you done these factorization algorithm, you taken F you take a divisor you divide. That is what is the idea now D is the divisor F. So, you have to find out the good divisor, that is very, very important. So, it is based on heuristics algorithms. So, that we are going to just discus some of the references, you can find out that in the fan out in the lectures.

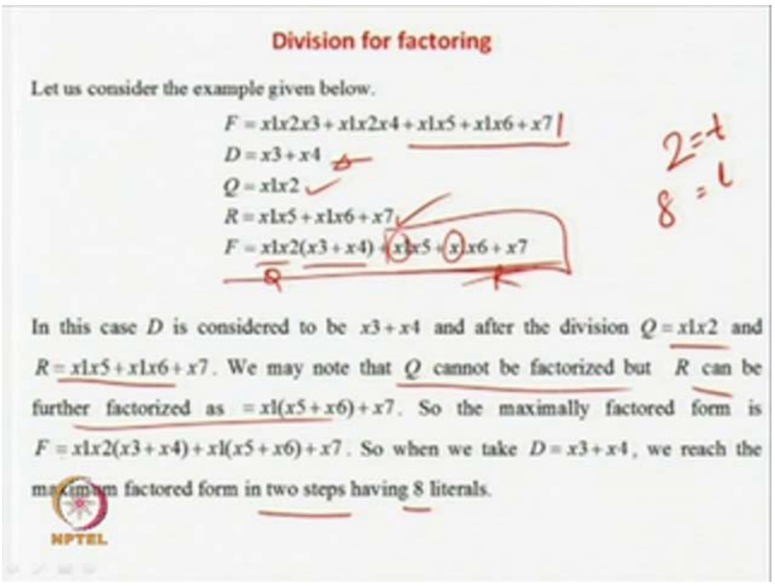
So, I -i mean uploaded in the lectures. So, there is you can find out some good algorithms, which can find out the how a good divisor can be obtain, but any way this is very complex procedure. So, we are not discussing in this lecture. What just you given idea, how you can find out some heuristic, which can determine in a good divisor.

So, now find out the good divisor, now what you do to do this division as exam as given to the examples, if your division is good find out that this reminder and the quotient, they cannot be again factored. But, if this is not the case then what you have to do again, you have to factorize R again, you have to factorize Q and infect, if you not taken a very good divisor also then again, you can able to factorize D and the process will keep on way, but for the general cases, D is m not be factorize over again.

But, it may happen then Q and R actually, factorized but, in worst case, if you are, if you are not a very good person at selecting D then you can use a D, which may again re factored. So, can you keep on factoring D Q N F, you get a you give at the time where, this expression cannot be factored again and you will find the values.

So, that is what I -i told you is given in this. So, you start with your you will find out the good candidate. So, then you actually, divide this then you get this one. So, if can again factorize, this Q and R and keep on doing till get factorization. So, that is the idea.

(Refer Slide Time: 48:04)



Division for factoring

Let us consider the example given below.

$F = x1x2x3 + x1x2x4 + x1x5 + x1x6 + x7$

$D = x3 + x4$

$Q = x1x2$

$R = x1x5 + x1x6 + x7$

$F = x1x2(x3 + x4) + x1x5 + x1x6 + x7$

In this case D is considered to be $x3 + x4$ and after the division $Q = x1x2$ and $R = x1x5 + x1x6 + x7$. We may note that Q cannot be factorized but R can be further factorized as $= x1(x5 + x6) + x7$. So the maximally factored form is $F = x1x2(x3 + x4) + x1(x5 + x6) + x7$. So when we take $D = x3 + x4$, we reach the maximum factored form in two steps having 8 literals.

Now, we will be in the whole things some examples. So, this is the F and then this be the D, Then if you repeated all these steps, you will find out that, this is the quotient and this is the reminder. So, in the examples, there are not going to show again how it comes, because it will be very easily find out found, you can very easily found out the the same procedure first, you take x 3 and find out the terms.

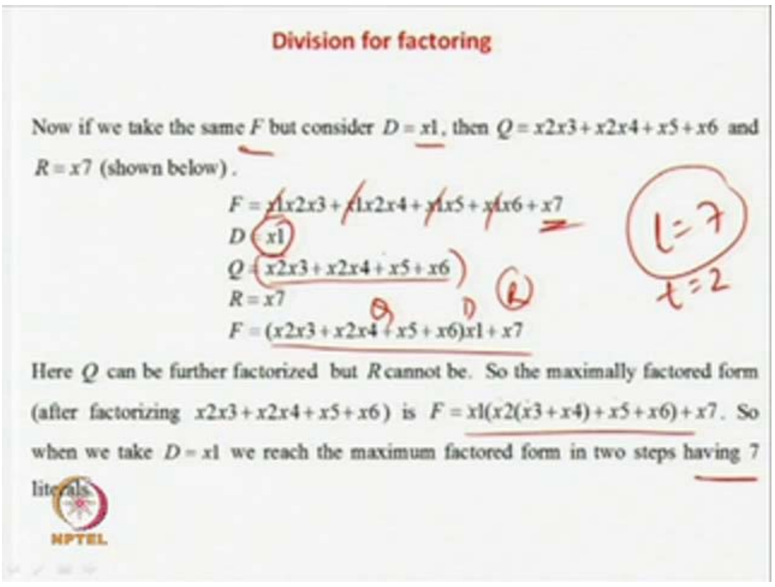
Then you take x 4 and you find the D 2 then you find out what the common terms then you then you actually, in this case x 3 plus x 4, you will find in this case, you will fin that x 1 and x 2 for the common terms over here then.

So, it will you are a expression will be x 1 dot x 2 dots in the plus 4 and these 3 terms like x 1 x 5 x 1 x is x term cannot included mean, these cannot be factored this divisor. So, this is your quotient and this is your reminder, in this case, we consider this one, that divisor is this. So, it is the reminder and the quotient is x 1 and x 2, we note that quotient can be factorize again of this, but R can be factorize.

So, in this case you can see that, you can see this cannot be a very good divisor. So, this is not the very good divisor, because using this divisor what happened, this quotient, you have got the reminder. Of course, which are the factored that is the good point or the bad point is that reminder can again be factorized it can easily factorize by use this x 1 out x 1 is can be done.

So, finally, we have 1 2 3 4 5 6 7 8 so, we have 8 literals and we reach in 2 step, because first you get this term and then again you have to factorize R. So, so time step required 2 times and number of literals is equal to how much 1 2 3 4 5 6 7 sorry, 1 2 3 4 5 6 7 8. So, 8 literal form this. So, this what I -i have obtain.

(Refer Slide Time: 49:49)



Division for factoring

Now if we take the same F but consider $D = x1$, then $Q = x2x3 + x2x4 + x5 + x6$ and $R = x7$ (shown below).

$$F = \cancel{x1}x2x3 + \cancel{x1}x2x4 + \cancel{x1}x5 + \cancel{x1}x6 + x7$$
$$D = x1$$
$$Q = \underline{x2x3 + x2x4 + x5 + x6}$$
$$R = x7$$
$$F = \underline{(\overset{Q}{x2x3 + x2x4 + x5 + x6})\overset{D}{x1} + \overset{R}{x7}}$$

$$\boxed{l = 7}$$
$$t = 2$$

Here Q can be further factorized but R cannot be. So the maximally factored form (after factorizing $x2x3 + x2x4 + x5 + x6$) is $F = \underline{x1(x2(x3 + x4) + x5 + x6) + x7}$. So when we take $D = x1$ we reach the maximum factored form in two steps having 7 literals.

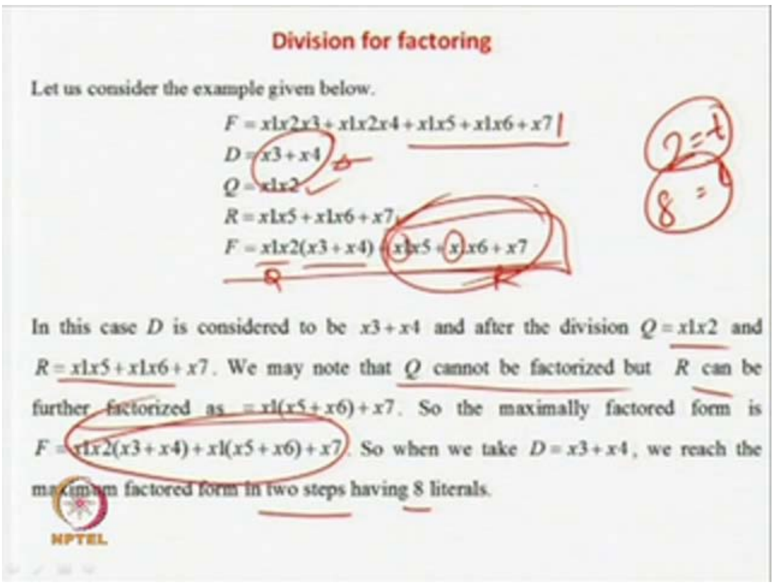
NPTEL

Now, you see same F, but now we have change the factor. So, we have just try to show you that how important D is to get a maximally factored form. So, now D is this. So, now, if you take x 1.

So, you can easily find out that, this one will be your quotient. But, the same term and reminder will be x 4, because you see x 1 is common over here, every time you have cut it of so, this one will be your quotient and x 7. So, this one is your quotient, this one is your D and this one the reminder. So, it is very obese that reminder cannot be factored D is a good, this is D, this cannot be factored, but you can see, you can be further factored.

So, be further factorize Q, you can find out that, this one is the answer. So, in this case how much 1 2 3 4 5 6 7. So, you have seven literals time, literal is equal to 7 and time step is equal to 2. So, if you consider this D time step is 2 and literal is 8 S in this case time step is to time remain save, what you have to decrease 1 literal. So, you can say that, D equal to x 1 is a better divisor compare to x 1 x 3 plus x 4. So, they can give you further divisor what you have see then.

(Refer Slide Time: 51:58)



Division for factoring

Let us consider the example given below.

$F = x1x2x3 + x1x2x4 + x1x5 + x1x6 + x7$

$D = x3 + x4$

$Q = x1x2$

$R = x1x5 + x1x6 + x7$

$F = \underbrace{x1x2}_{Q}(x3 + x4) + \underbrace{x1x5 + x1x6 + x7}_{R}$

In this case D is considered to be $x3 + x4$ and after the division $Q = x1x2$ and $R = x1x5 + x1x6 + x7$. We may note that Q cannot be factorized but R can be further factorized as $= x1(x5 + x6) + x7$. So the maximally factored form is $F = x1x2(x3 + x4) + x1(x5 + x6) + x7$. So when we take $D = x3 + x4$, we reach the maximum factored form in two steps having 8 literals.

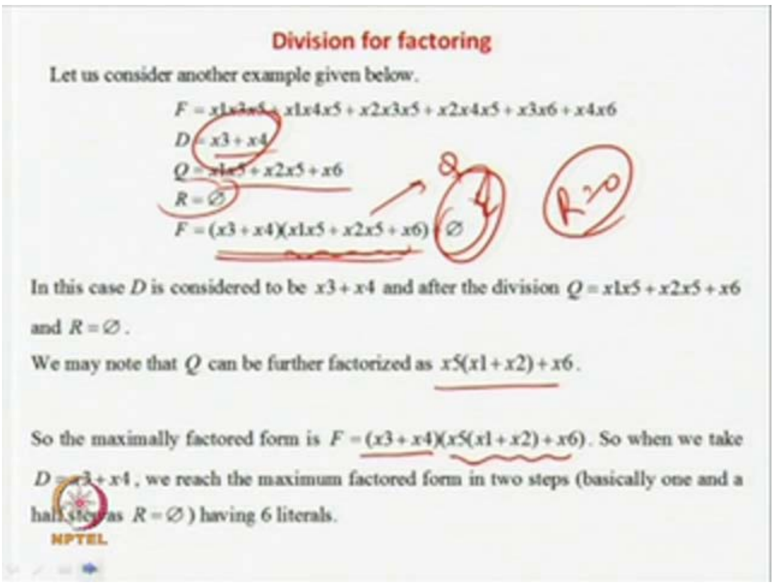
What you have see is that base on divisor, what is the idea based on the divisor the time taken 2 factorize, the function also changes and quality of function also changes, now we take a another example, we will now D equal to 2. So, if take D equal to 2 2, you will find out, if do that, we will find out the quotient is this and reminder is this. So, in this case you can find out the that, after the divisor this one is the quotient and we may not that Q can be factorize at this one R can also be factorize as this one.

So, now, with this divisor factorize, you can, you have situation. So, you can also be factored all can also be factor. So, in this case what will happen Q could not factor sorry, R could not be factored, but Q could be factor and in the first stage what the happen ah Q could not be factored could be factored. So, you see depending upon this choice of reminder, what happen what is happen in this sorry. So, depending upon the choice on the divisor also what is happening. So, different type of situation is arising both Q and R factored R can be factored Q can be factored and so for. If this x 2, so is the quotient.

This is the divisor in this case both can be factored. So, now, the solution is this one. So, this is the final solution. So, in this case you have 1 2 3 4 5 6 7 8. So, you have 8 literals and again 2 time steps. So, 8 literals and and time step is also equal to 2. So, you can say that 2 is the bad not a good good factor now divisor, but 2 is the best divisor then x 3 plus x 4, because in x 3 plus x 4, you required 2 time step and literal is 8.

So, in this case this is the same 8 and 2 things, but here after first step, you have to factorize both Q and R again, but if you take x 3 plus x 4 then then time, it have almost same, but actually in the second step second step you you are factoring only R you are not factoring Q. So, you can say that, the best divisor in this case amount the 3 is this one. So, where we have 7 literals 2 times steps. And you need to factorize only the Q in the same again second one x 3 plus x 4 where the literals is 8 and time is 2, but you are factoring only R and the where most 1 is x 2. Where you have I -i mean, you have taken time step at a 2, but you have taken factorize Q and R both and again the number of literal uses 8.

(Refer Slide Time: 53:09)



Division for factoring

Let us consider another example given below.

$F = x1x3x5 + x1x4x5 + x2x3x5 + x2x4x5 + x3x6 + x4x6$

$D = x3 + x4$

$Q = x1x5 + x2x5 + x6$

$R = \varnothing$

$F = (x3 + x4)(x1x5 + x2x5 + x6) + \varnothing$

In this case D is considered to be $x3 + x4$ and after the division $Q = x1x5 + x2x5 + x6$ and $R = \varnothing$.

We may note that Q can be further factorized as $x5(x1 + x2) + x6$.

So the maximally factored form is $F = (x3 + x4)(x5(x1 + x2) + x6)$. So when we take $D = x3 + x4$, we reach the maximum factored form in two steps (basically one and a half steps as $R = \varnothing$) having 6 literals.

This one is the most bad and the worst amount the divisor. So, let us take an the example. So, we have showing different examples, that how D is very good how D is quality of the algorithms or the flow of multilevel implementation. So, this is another F and they have taken a factored divisor call D. So, in this case you take this D will find out that, this is your quotient reminder is 0.

So, you have getting something like this, in this case we can call that here, D in nothing, but D is factored because this can very well means I -i mean, because it can factorize F to the generated R. So, this on is the case, we know that now, we can have this is actually quotient and further factorize into this. So, maximally factored from you can find out for this one will be x 3 plus x 4 and this for a, this is this part is actually, Q.

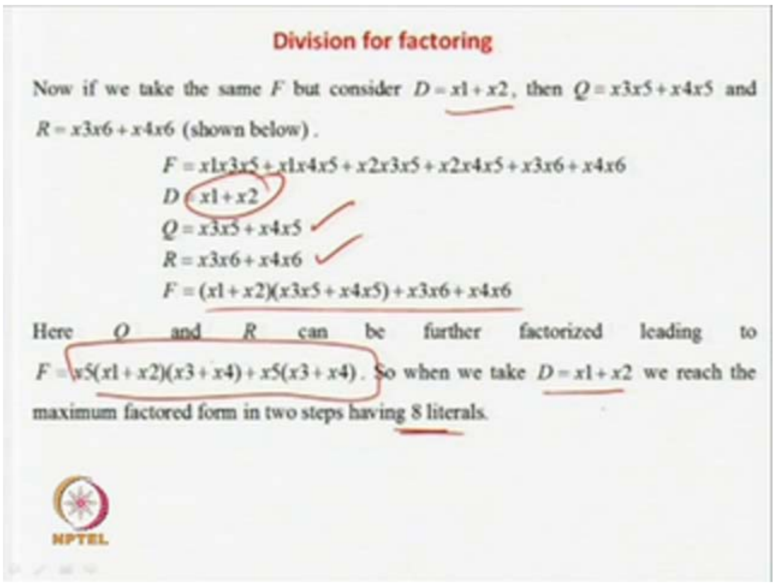
So, can factorize this into this. So, when we take ah this one, you get a maximally factored form into steps having 6 literal. So, in this case 1 2 3 4 5 and 6 literals are there. So, that is what is mean save, this is not an another example, you can say that is the type of divisor, we want to find out the where is good divisor and bad divisor.

So, you can think that, there is very good divisor, because at least you can generate R, which is reminder is 0, if you could reminder reminder as a 0, very good divisor, there very good divisor. You have found out, you can fully factorization a expression reminder, even in zero that means, there is no Q were.

So, have made a lot of saving in the implementation, now you can again find out that can again re factored that is different story. So, if you can find out that divisor, which can entire a divide their function without generating, you can say with a very good divisor, because what because, you are not generate the reminder that means so, number of cubes are reduce. So, your implementation is minimize what just like prime number, you cannot I -i there is no divisor, which can divide the number accept number itself.

So, I -i mean obese the if this is a cannot generate to this is equal to this is equal to D, that is a very foolish way, we will think about it. So, just like some prime number such that, which cannot be further factorize without 1 and the number itself. Similarly, there can be some function for you can never find out a divisor where, the reminder will be 0 accepting the function itself, but cannot the obese the this is the divisor over divisor over, which is very well known. So, if some functions are there for you can find out the divisor can actually, give you what do I -i say.

(Refer Slide Time: 55:30)



Division for factoring

Now if we take the same F but consider $D = \underline{x1 + x2}$, then $Q = x3x5 + x4x5$ and $R = x3x6 + x4x6$ (shown below).

$$F = x1x3x5 + x1x4x5 + x2x3x5 + x2x4x5 + x3x6 + x4x6$$
$$D = \boxed{x1 + x2}$$
$$Q = x3x5 + x4x5 \checkmark$$
$$R = x3x6 + x4x6 \checkmark$$
$$F = \underline{(x1 + x2)(x3x5 + x4x5) + x3x6 + x4x6}$$

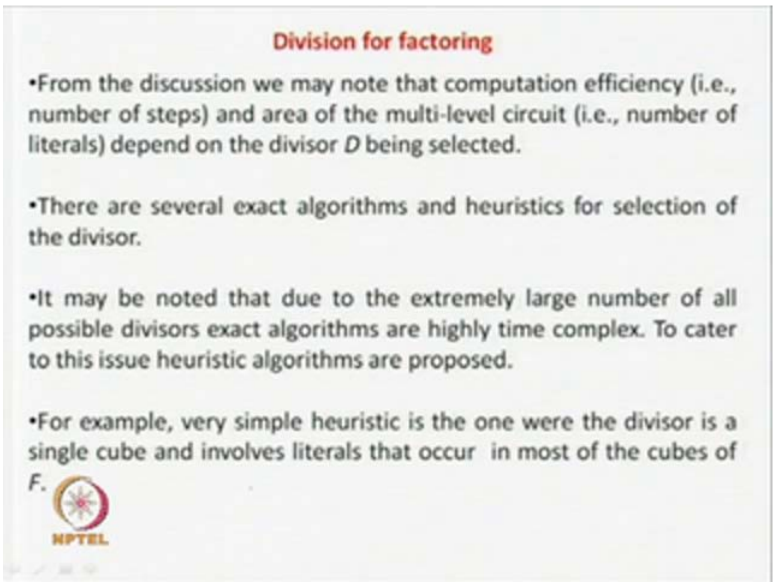
Here Q and R can be further factorized leading to $F = \boxed{x5(x1 + x2)(x3 + x4) + x5(x3 + x4)}$. So when we take $D = \underline{x1 + x2}$ we reach the maximum factored form in two steps having $\underline{8}$ literals.

The minimum representation I -i mean remain remain 0, you can consider is very good divisor. Now, you see so, this was her 6 literals. This case now in this case, now if you take same F now you consider x 1 plus x 2 as a divisor. So, we will find out that, this is the quotient and this is the reminder.

So, now you have seen that x 1 plus x 2 is not a factor. But, in this case x 2 plus x 4 was a factor. So, what happens now in this case, this is the quotient and this is reminder. So, if you represent F. So, are going to get something like this.

So, in this case, you can find out that would can be factorized and finally, we have this solution. In this case, these the divisor so, maximum factored form is I -i mean 2 literal 8 literals and time step is 2. So, it is 1 2 3 4 5 6 7 and 8 literals and time step is 2, in this case the number of literal is 6, 1 2 3 4 5 6. So, that is why.

(Refer Slide Time: 56:22)



Division for factoring

•From the discussion we may note that computation efficiency (i.e., number of steps) and area of the multi-level circuit (i.e., number of literals) depend on the divisor *D* being selected.

•There are several exact algorithms and heuristics for selection of the divisor.

•It may be noted that due to the extremely large number of all possible divisors exact algorithms are highly time complex. To cater to this issue heuristic algorithms are proposed.

•For example, very simple heuristic is the one were the divisor is a single cube and involves literals that occur in most of the cubes of *F*.

If you find out a divisor actually, which is which can generate the reminder equal to 0, you consider that is the very good kind of divisor. So, no we have seen that, once you can find out the very good divisor and the problem is very simple. So, what to have to do that is this algorithm step you have already seen that is in simple in a step seen that what do you do you just generate the first cubes of their over here.

Factorize the Q, which have then by D 1 D 2 D 3 and then final common factors common literals and then you again generate generate the quotient and the reminder again, you try to find out whether the quotient and the reminder factored and you keep on doing it to reach a state where, nothing can be factorized over again.

So, that is what is going to tell you that, we have gone for the most factorize, but the most difficult problem is that, how can you given a function, how can you find out the good divisor. So, that is a very difficult problem and we are not going to elaborate in this, but we can say that, there are lot of exact algorithm lot of heuristics, which can you can find out, if you if you you can find out the references given in the hand out that, you can read, but you can find that why can be very complex, because if see that for the 2 level implementation, there was a some some solutions.

The solutions spaces are is what set of the primes. That is solution space, but in this case you have you have staring of the subset prime, because you have to go minimal level S O

P minimization minimize form, that is the subset of prime and from there you have to find out that what what I -i mean go for a multilevel factors form, that is there again.

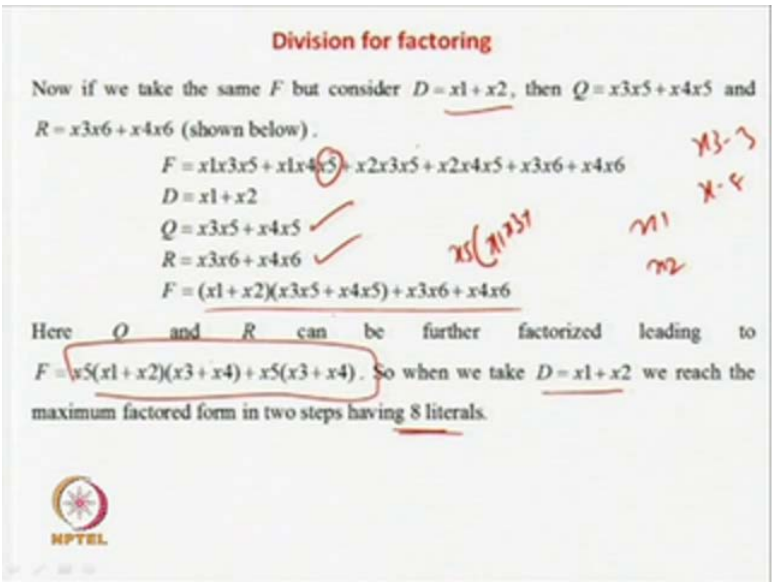
Now, just I -i mean I -i mean not going in elaborate, but just think of that, if if subset sum that is given some prime implement, you have to find out subset of them, that is number prime implement, that is solution space, but in this case you can see different depending upon different type of factors. You can see that a different type representation that are coming up.

So, you can see you just realize and think how the be the solution space, the solution space in case multilevel implementation is much much larger in the 2 level implementation two level implementation is nothing but, the power set of the prime , but in this case what is happening, it is it given as different type of factor.

So, factor can be n where n number of different, you can generate a factor and for each of the factor, you can find out, you can have just see how different factorization procedure . So, you can think of the solution bases extremely extremely large compare to 2 level implementation. So, that is why the algorithm of the heuristics to find out the good divisor is also a very difficult problem.

So, whenever the solution spaces small then your algorithm are simple, where are the solutions spaces are very very high, the algorithms are complex, that is why finding out a good divisor is a very difficult problem and we are nit going to elaborate, in this lecture what you can think very simple heuristics can be a divisor is the single Q, you can that I -i always take divisor single, Q then x plus y or something like this and involves literals that are occur then in most of the cubes of f.

(Refer Slide Time: 59:05)



Division for factoring

Now if we take the same F but consider $D = x1 + x2$, then $Q = x3x5 + x4x5$ and $R = x3x6 + x4x6$ (shown below).

$$F = x1x3x5 + x1x4x5 + x2x3x5 + x2x4x5 + x3x6 + x4x6$$
$$D = x1 + x2$$
$$Q = x3x5 + x4x5$$
$$R = x3x6 + x4x6$$
$$F = (x1 + x2)(x3x5 + x4x5) + x3x6 + x4x6$$

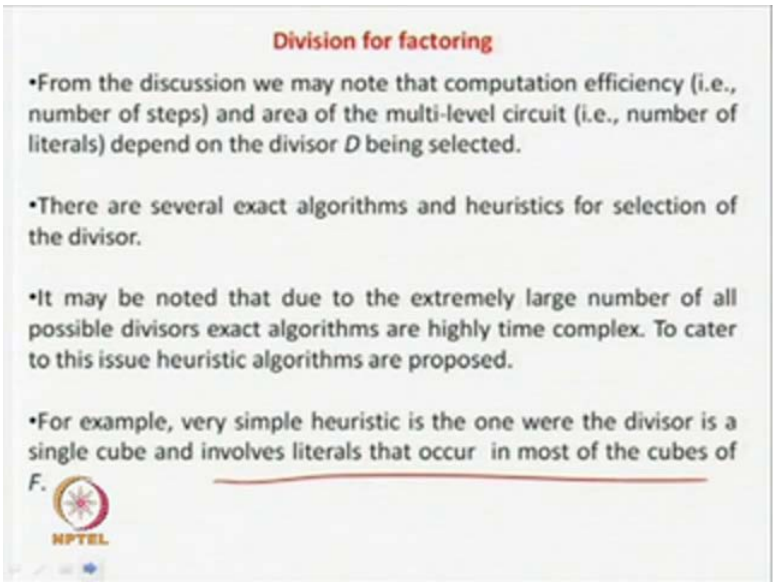
Here Q and R can be further factorized leading to $F = x5(x1 + x2)(x3 + x4) + x5(x3 + x4)$. So when we take $D = x1 + x2$ we reach the maximum factored form in two steps having 8 literals.

x3-3
x-4
x1
x2
NPTEL

So, just go for this, if you just give you example I -i want to find out a divisor of this one. So, I -i will not take divisor like x 1 plus x 2 or something like that, I -i will this either x 1 or x 2 sum up to x 6 1 of them, now which I -i will take I -i will take the literal, which is common to most of that like in this case 1 2. So, x 1 is in 2 places x 3 is 1 2 3 x 1 is x 3 is in 3 different places. X 5 is in 1 2 4 is x 5 is a 5 cubes.

Similarly, x 6 is in 1. So, you can say that I -i will take x 5 as a because x 1 appearing over here a very over here up to the over here over here. So, I -i take x 5 then going to get as x 1 plus x 3 sorry, x 1 x 3. I –i will get x 1 x 3 plus x 1 x 4 x 2 x 3 x 2 x 4 and so, for. So, that is if I -i take the literal, which id the common in most of cubes then, I -i can able I -i can able to reduce, I -i mean factorize the expression in a most.

(Refer Slide Time: 01:00:04)



Division for factoring

•From the discussion we may note that computation efficiency (i.e., number of steps) and area of the multi-level circuit (i.e., number of literals) depend on the divisor *D* being selected.

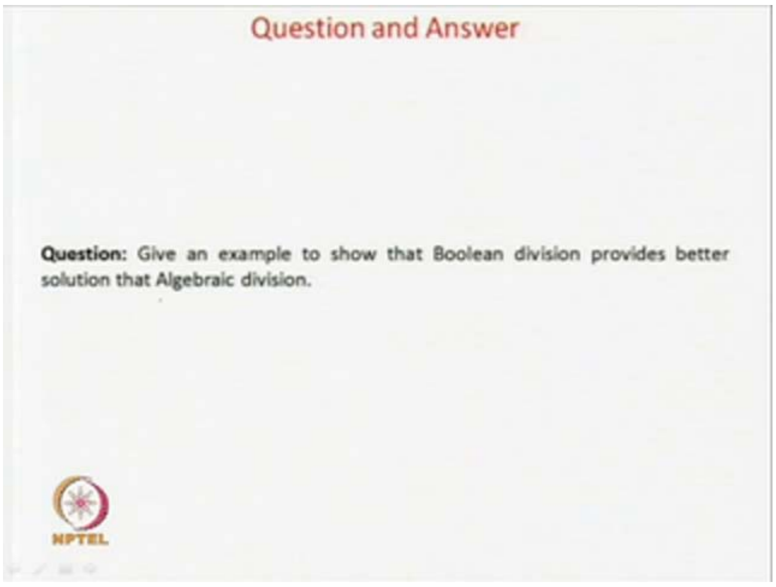
•There are several exact algorithms and heuristics for selection of the divisor.

•It may be noted that due to the extremely large number of all possible divisors exact algorithms are highly time complex. To cater to this issue heuristic algorithms are proposed.

•For example, very simple heuristic is the one were the divisor is a single cube and involves literals that occur in most of the cubes of *F*.

So, this is the one very simple heuristic way, you can think, but exact algorithms and heuristics are more complex. So, a with this, we actually closing this discussion and we are coming to question answer session.

(Refer Slide Time: 01:00:17)



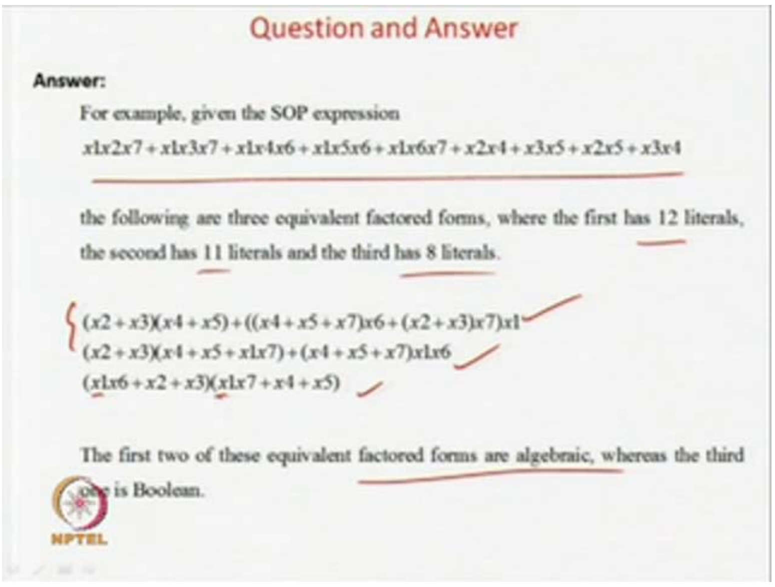
Question and Answer

Question: Give an example to show that Boolean division provides better solution that Algebraic division.

So, we just say that given example, we show that the Boolean divisor procedure better solution have at prime and that is very important now, till, now we discussion what to have found of we have discus that, all we have done is basically nothing. But, we are all talking about different type of expression divisor etcetera all are all all algebraic divisor.

So, what are the idea that algebraic divisor, that you have 2 terms, we should not x 1 here, on 1 that is the idea and if you allow this then, you will become a Boolean division and infect, Boolean be give you better solution, then algebraic division, but Boolean, but Boolean divisor was more complex then algebraic divisor, that is we have not gone into this, but just to give you the idea.

(Refer Slide Time: 01:00:56)



Question and Answer

Answer:

For example, given the SOP expression

$x1x2x7 + x1x3x7 + x1x4x6 + x1x5x6 + x1x6x7 + x2x4 + x3x5 + x2x5 + x3x4$

the following are three equivalent factored forms, where the first has 12 literals, the second has 11 literals and the third has 8 literals.

$(x2 + x3)(x4 + x5) + ((x4 + x5 + x7)x6 + (x2 + x3)x7)x1$

$(x2 + x3)(x4 + x5 + x1x7) + (x4 + x5 + x7)x1x6$

$(x1x6 + x2 + x3)(x1x7 + x4 + x5)$

The first two of these equivalent factored forms are algebraic, whereas the third one is Boolean.

That, but is the case. So, if this is your some f, then three equivalent factor forms. First is literal then second 1 11 literals and third one has 8 literals, this is the first, this is second and this is the, so you can observe that, these 2 are algebraic and the third one is a Boolean, because you can see x 1 x 1 is common over here.

So, I -i mean this is a very common topics and we are not going into details into this course, but the idea is that, you can easily you can you can verify, if you are, if you go in to the depth that Boolean expression I -i mean, Boolean factor from R, further minimize the that to algebraic form. But algebraic division and algebraic is minimization of the factorization is a more complex procedure.

So, we have not gone into depth of this, that is why we just given you overview and infect multilevel implementation is a very complex topic. So, in this this course, we have tried to give you the exposure in a very minimal way just to give you an idea just that given a 2 level form what is the problem and how we can go and approach in a multiple level implementation, we have not gone any amount of depth by any means in this multi level implementation.

So, there is lot of words, which is been done and a lot of word still is going on in multilevel implementation, that is what is practically, require for implementation, this is type of the eyes for multilevel implementation, we have given you and the we have also show you that algebraic with factorization, if you compare with Boolean Boolean gives a much better solution. But, again this is a very more complex though, you have not even discussion how to do Boolean level Boolean level factorization for the in that case. So, with this, we close this lecture.

(Refer Slide Time: 01:02:26)

Thank You

And also we come to the end of the design module. So, in the design module, what we have seen, we have seen that, we start of the design specification and from design specification, we go for high level synthesis. For high level synthesis, we go for Boolean synthesis, first you for multilevel sorry, first you for 2 level implementation and if is the sequence circuit then you have go for state minimization and state encoding and then you go for minimization, if is combinational circuit, you can directly go for 2 level minimization.

And after the 2 level minimization is done, we go for multiple implementations and the final at least for the gate level implementation goes on for what you can say fabric location what.

What before you go on a fabrication, we have to se that all the transformation, we have done, specification to high level deigns, high level design to Boolean level design 2 level from 2 level to multiple level then it should be all equivalent. That is there should not be any changes specification in between this stuff.

So, the next module, we are going to see formal verification take needs that will that is the design A is equivalent design B, design B equivalent in design C. So, you can say that with the final design is out, you can claim that final design is same as is equivalent to my specification or they needs my specification. So, in the next module of the course at the this course is design and tests, in the next module, we will see about verification.

Thank you.