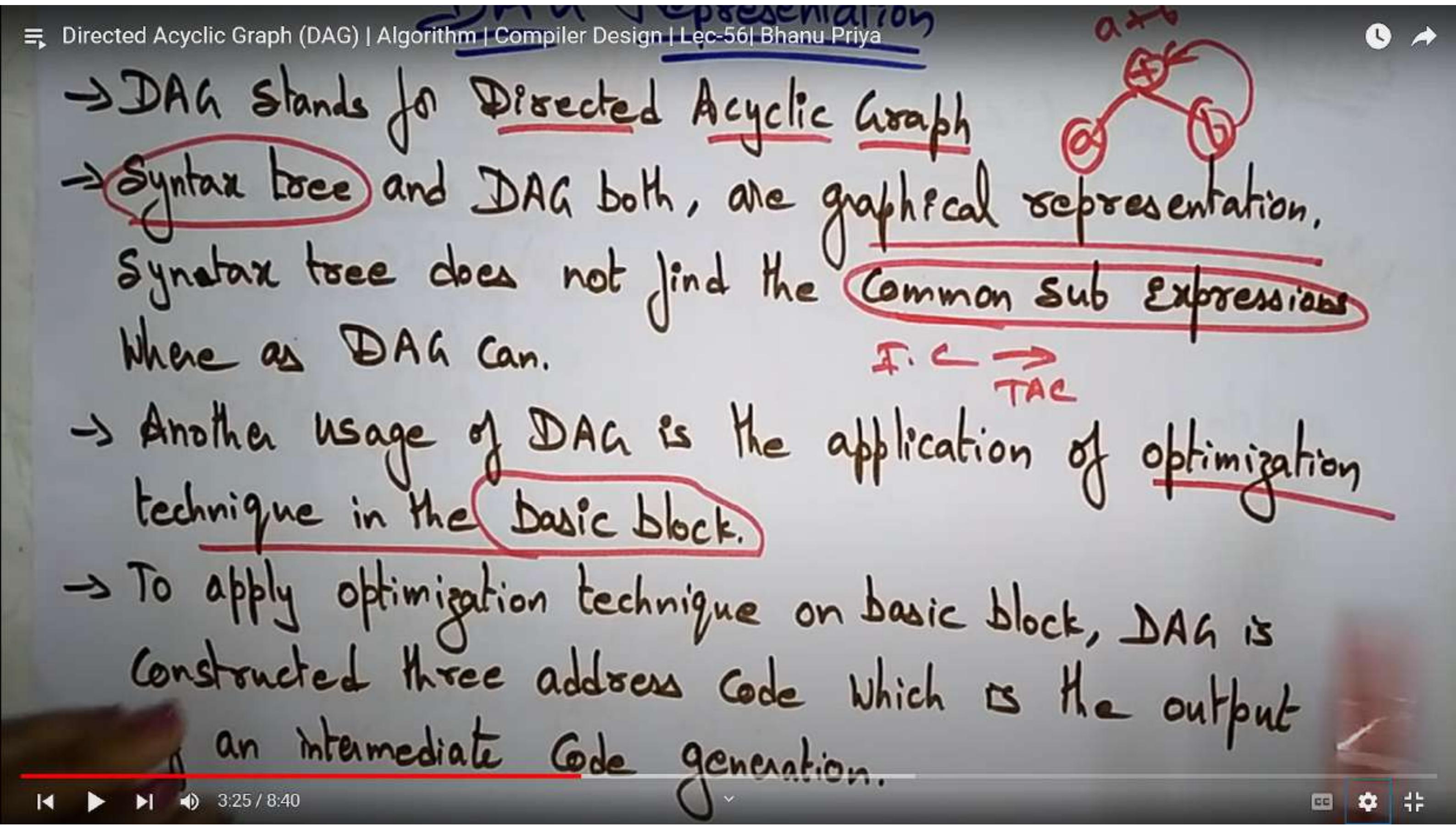

Directed Acyclic Graph (DAG) | Algorithm | Compiler Design | Lec-56| Bhanu Priya
3:25 / 8:40
a+b
+
a
b

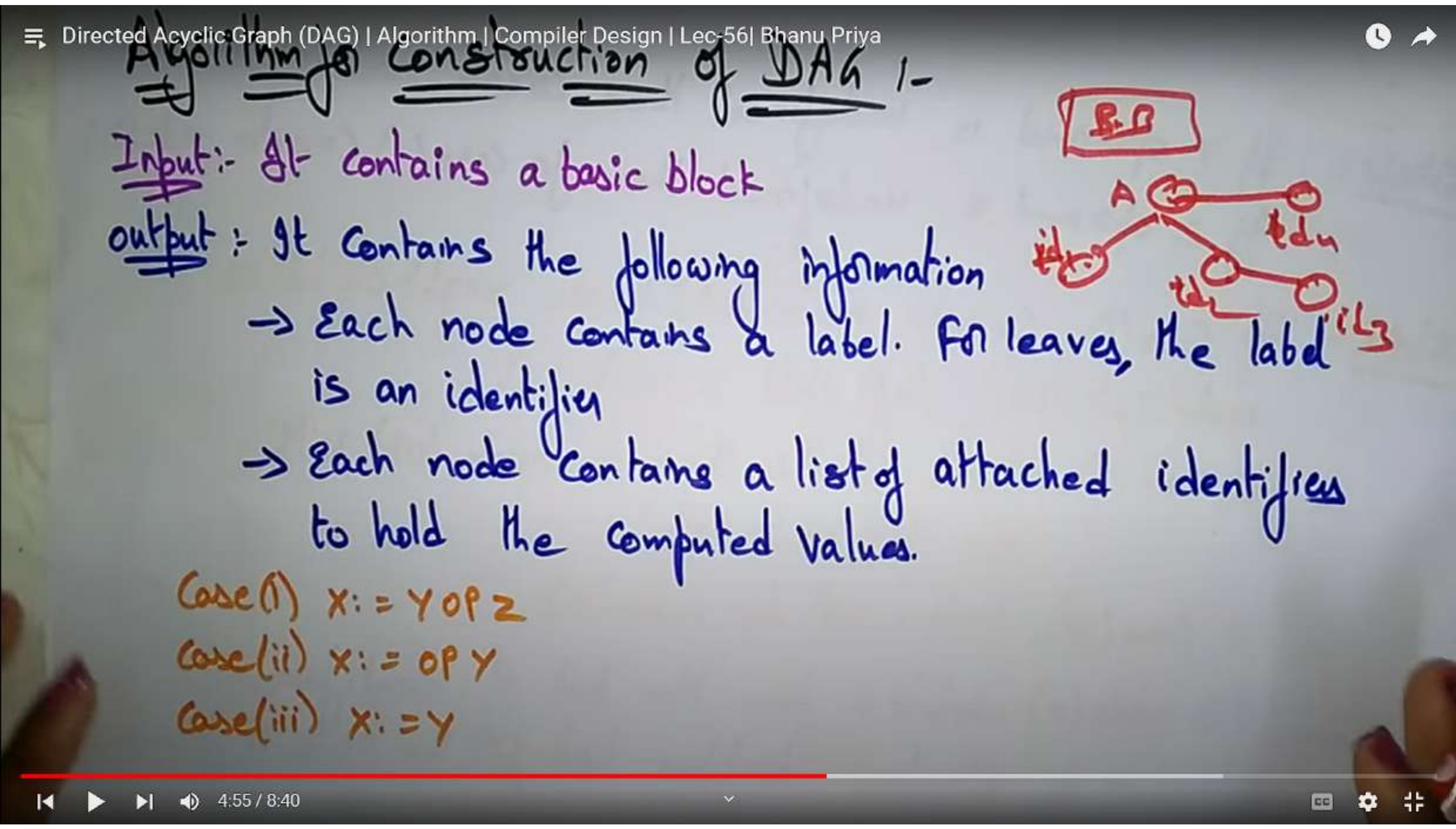
→ DAG stands for Directed Acyclic Graph

→ Syntax tree and DAG both, are graphical representation. Synatax tree does not find the Common sub Expressions where as DAG can.

I.C → TAC

→ Another usage of DAG is the application of optimization technique in the Basic block.

→ To apply optimization technique on basic block, DAG is Constructed three address code which is the output of an intermediate Code generation.



Directed Acyclic Graph (DAG) | Algorithm | Compiler Design | Lec-56| Bhanu Priya

Algorithm for Construction of DAG :-

Input :- It contains a basic block

output : It Contains the following information

→ Each node contains a label. For leaves, the label is an identifier

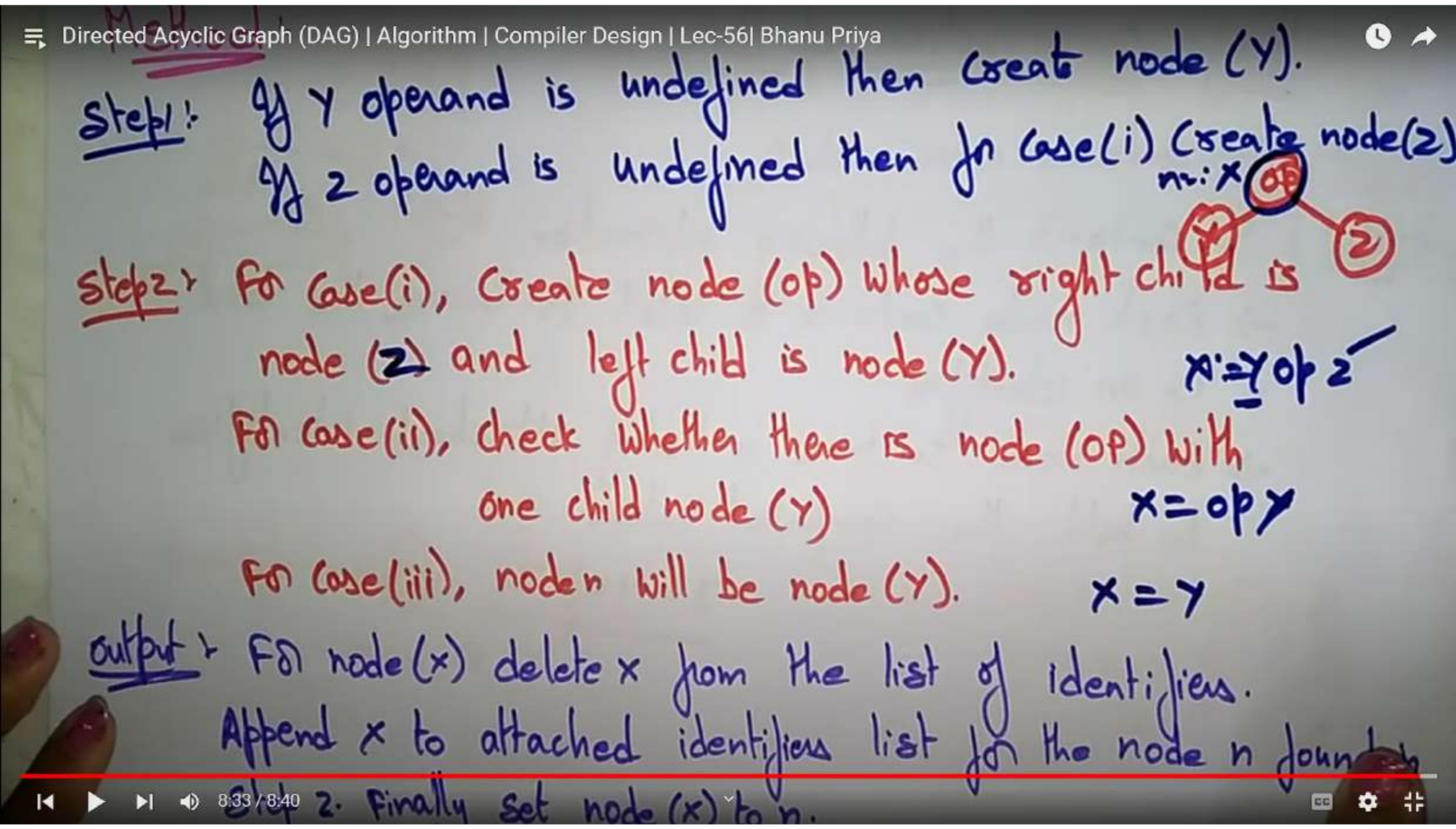
→ Each node contains a list of attached identifiers to hold the computed values.

Case(i) x: = y op z

Case(ii) x: = op y

Case(iii) x: = y

4:55 / 8:40



Directed Acyclic Graph (DAG) | Algorithm | Compiler Design | Lec-56| Bhanu Priya

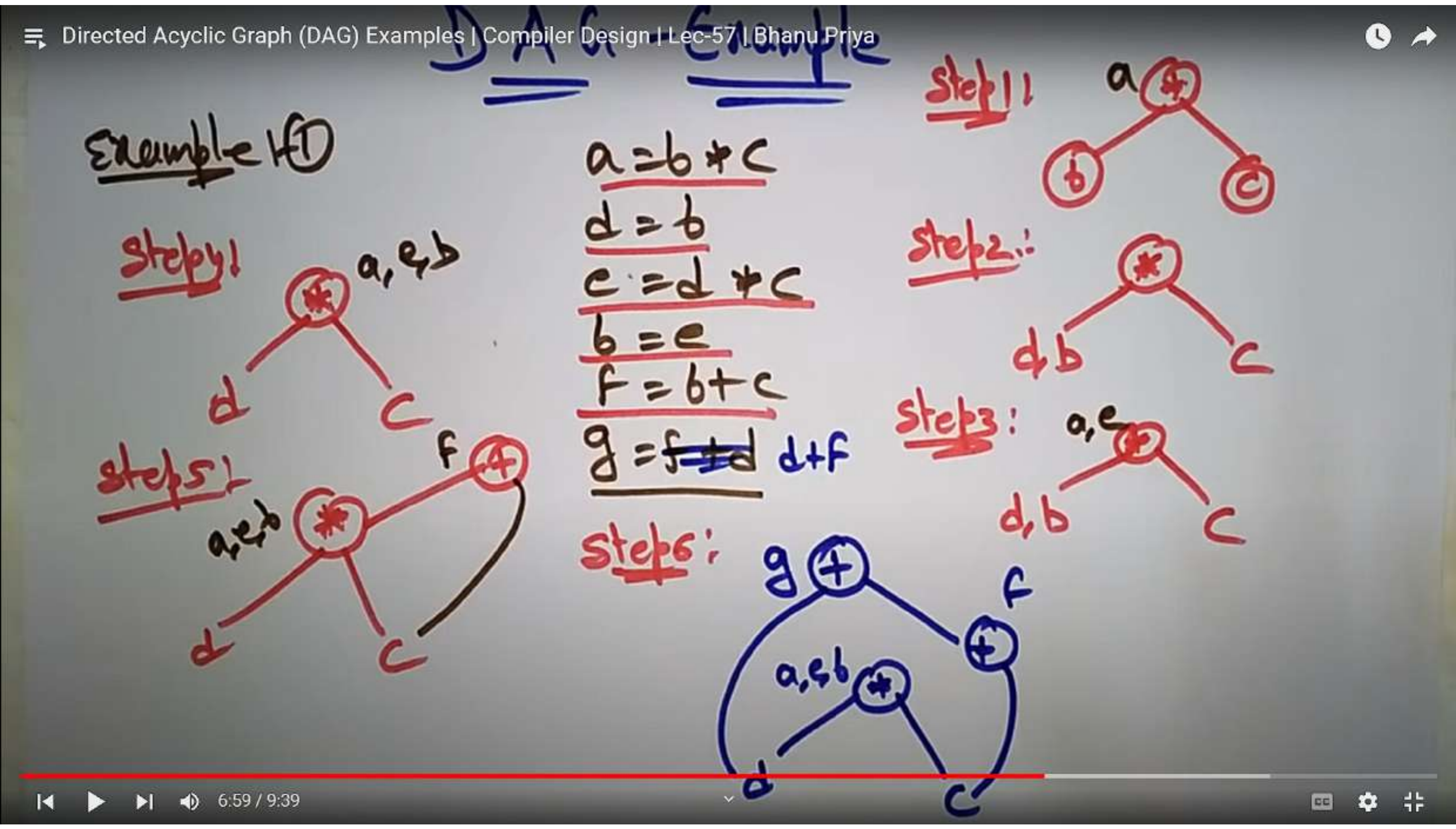
Step1: If Y operand is undefined then create node (Y).
If Z operand is undefined then for case(i) create node(Z)

Step2: For case(i), create node (op) whose right child is node (Z) and left child is node (Y). $X := Y \text{ op } Z$

For case(ii), check whether there is node (op) with one child node (Y) $X = \text{op } Y$

For case(iii), node n will be node (Y). $X = Y$

output: For node (x) delete x from the list of identifiers.
Append x to attached identifiers list for the node n found in step 2. Finally set node (x) to n.

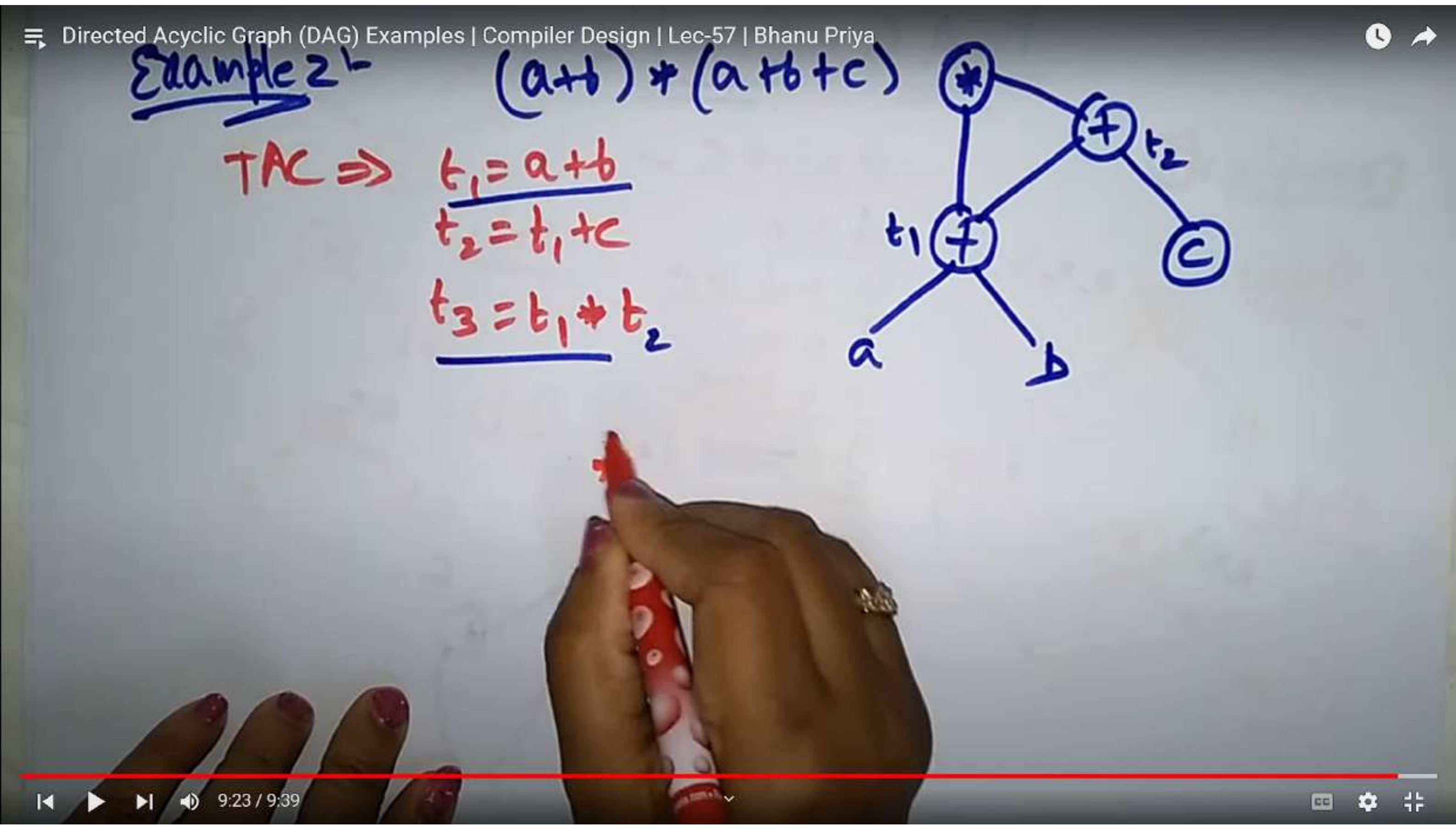


Directed Acyclic Graph (DAG) Examples | Compiler Design | Lec-57 | Bhanu Priya

DAG Example

Example 1-①

a = b * c
d = b
e = d * c
b = e
f = b + c
g = ~~f + d~~ d+f


Directed Acyclic Graph (DAG) Examples | Compiler Design | Lec-57 | Bhanu Priya

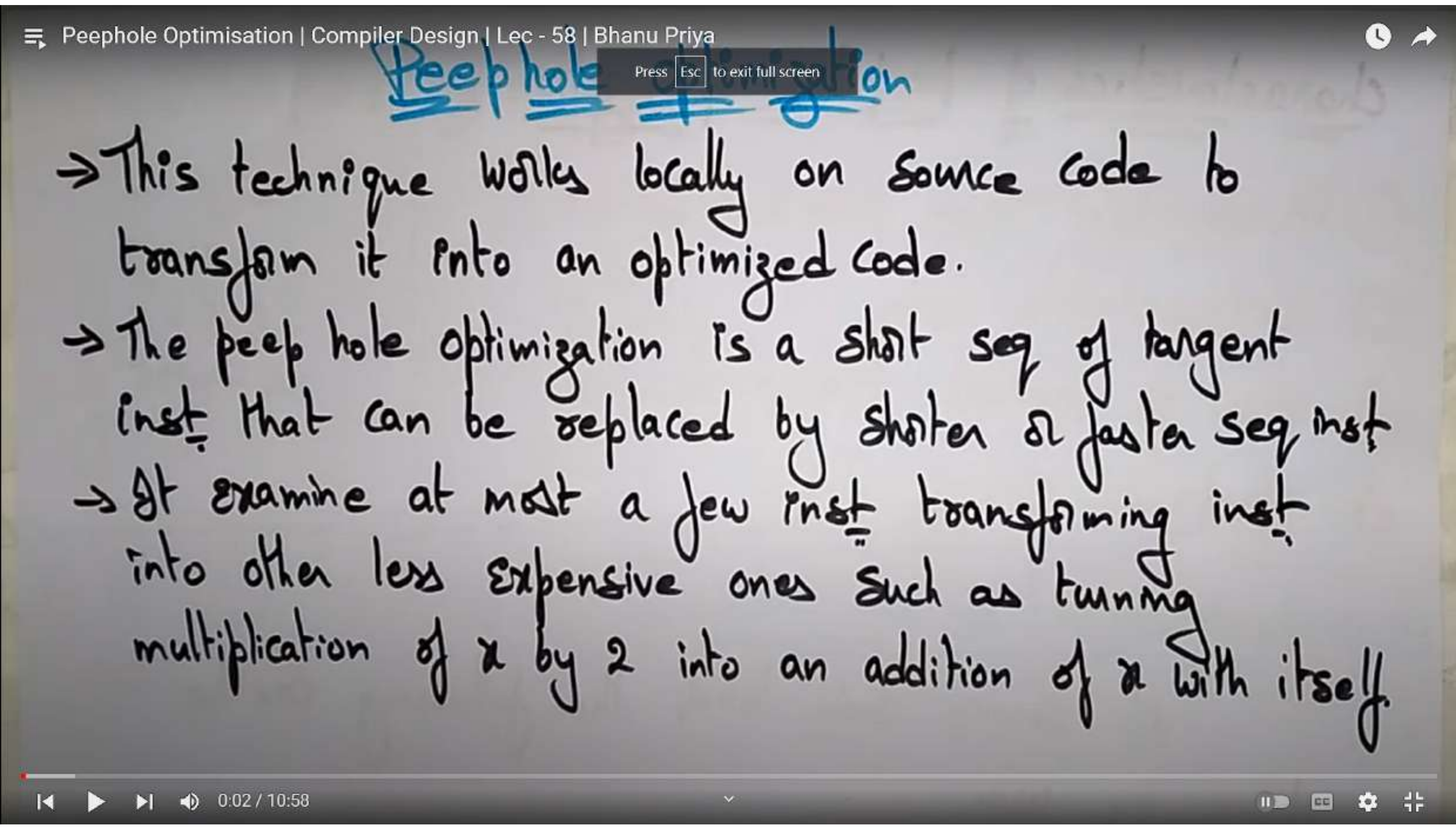
Example 2

$(a+b) * (a+b+c)$

TAC $\Rightarrow$ $t_1 = a+b$

$t_2 = t_1 + c$

$t_3 = t_1 * t_2$


Peephole Optimisation | Compiler Design | Lec - 58 | Bhanu Priya

Press Esc to exit full screen

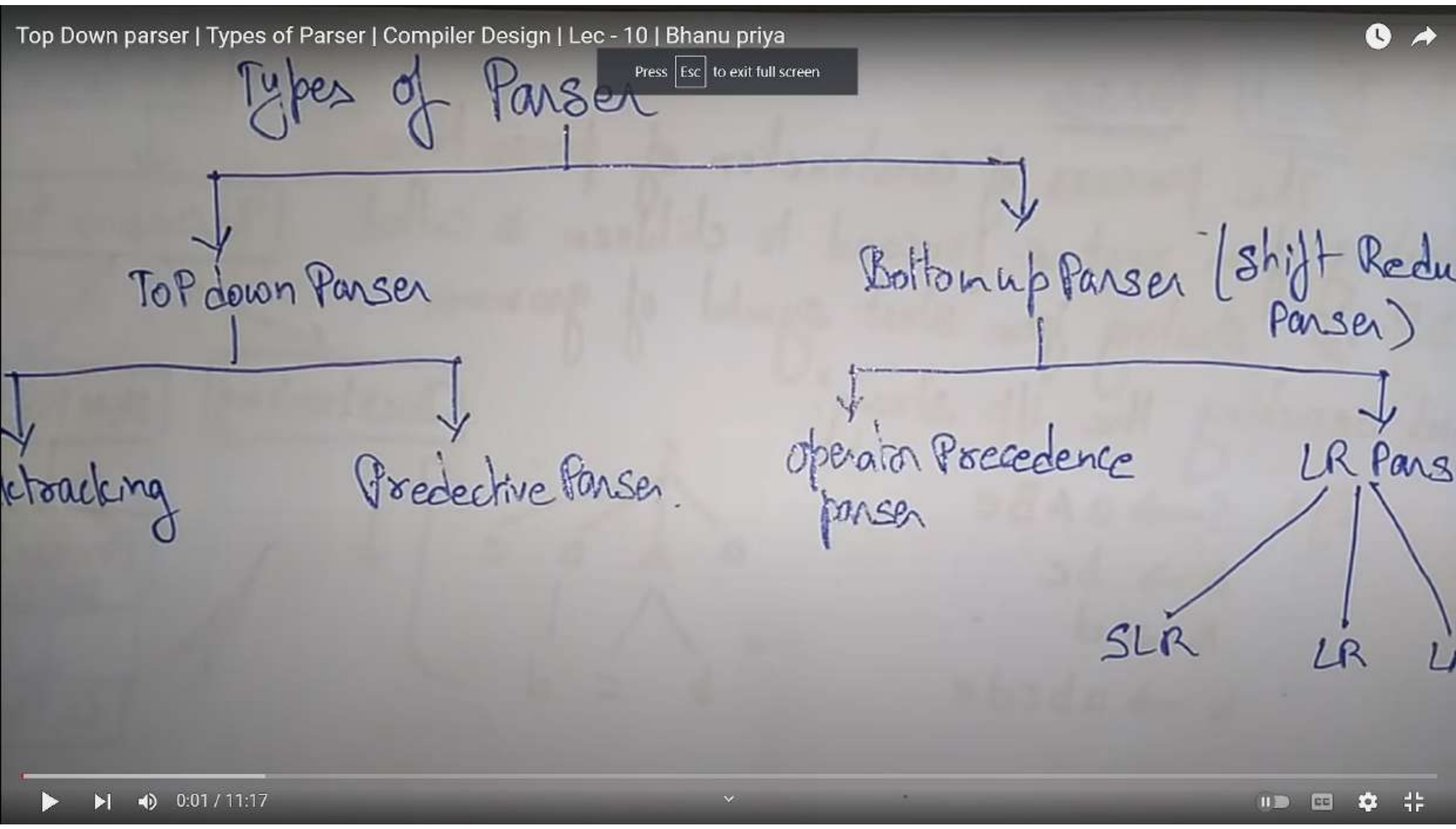
Peep hole optimization

→ This technique works locally on source code to transform it into an optimized code.

→ The peep hole optimization is a short seq of tangent inst that can be replaced by shorter or faster seq inst

→ It examine at most a few inst transforming inst into other less expensive ones such as turning multiplication of x by 2 into an addition of x with itself.

0:02 / 10:58


Top Down parser | Types of Parser | Compiler Design | Lec - 10 | Bhanu priya