

Optimization Techniques for Digital VLSI Design
Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Lecture – 22
Verification: ADD based verification, HDD based verification

So, welcome again to the course on Optimization Techniques for VLSI circuits and at present we are going through the module on verification.

(Refer Slide Time: 00:34)

Course Plan

Module 5: Verification

Lecture 1: LTL/CTL based Verification

Lecture 2 and 3: Verification of Large Scale Systems

- BDD based verification
- ADD, HDD based verification

Lecture 4, 5: Symbolic Model Checking

Lecture 6: Bounded Model Checking

So, in the last two lectures set we have seen that basically the major problem for any kind of verification or handling any kind of systems, where actually we have to go for a state space model then actually the size of the states space becomes so large, because it actually tries to blow up in the order of exponential complexity the number of system variables. So, that was the major problems of the bottleneck for in for testing as well as their verification.

So, whenever something you have to do or some automated procedure you have to execute on a system, where you have to model the system such that each such that each state or the set of states use to represent the system is in terms of the state variables, then actually it may blow up with in which state exponential problem and everything was

down that you can solve only for toy examples and many in our course such we know if some optimization.

So, that we can handle very large systems, so the last two lecture series we have seen that basically if you say shear state base or a simple final state machine or a state with exponential number of combination of states are there. Basically it correlates something to the binary decision tree that for a given set of variables, Boolean variables you have to go a exponentially in the order of 2 to the power n in the number of variables; like we have 1 left chain for true right chain for sorry left chain for false right chain for true and you have to keep on doing it will all the labels are explode.

So, that was the basic problem of explicit enumeration then in the two lectures, last two lecture series we have seen a very important data structure call binary decision diagram ; there we are seen basically we can compress, but eliminating all sort of redundancies and without any loss or in any quality of solution. Basically if you take if you say that I am a system there is a 100 state variables actually number of states should be much much less than 2 to the power 100, because there are lot of combinations we can never used in real practice.

So, basically BDD tries to leverage on that fact then eliminate all the all the redundant parts I, we actually explicitly model only the set which are absolutely require there is there is no redundancy and then we can see that the compression level is more than 99 percent or above, then we have seen some additional more advance versus like additional means arithmetic decisions diagram, high level decision diagram, which can even handle much larger circuit because they work at a higher level of abstraction.

But now the next 3 lectures basically that symbolic model checking and bounded model checking, here what we are going to do because as we told in the last class if you remember that all though HDD, but or arithmetic decision diagram can handle very large circuits because they go for modeling are the abstract level that is the RTI level. But still the research is going on there how can you go for good can only you can representation, how can you find out good model checking or that is the labeling algorithms.

If we are going in the model this systems as in terms of high level decision diagrams, whether not in the form of a bookie automata or a final state machine because, all the algorithms which we are seen in lecture 1 in case of CTL and LTL model checking or

rather if you tell me the ATPC in terms of the algorithm; we are very well the algorithms are very well develop if you consider the gate level modeling or explicit modeling or you can say the binary bit level modeling.

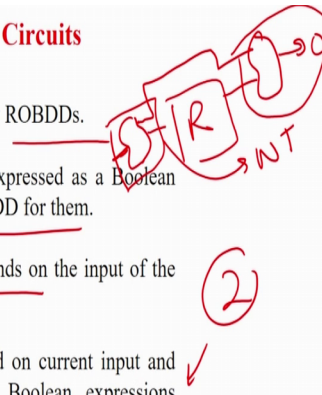
Once you go to higher levels of abstraction such algorithms are still in the development piece. So, we are we are not going to focus too much on again on HDD or arithmetic decision diagram you use them for verification because, this is more or on a research topic whether what you will try to do is that whether you will again go back to the bit level because, that is where all the strong algorithms of model checking are available, but then will see then you already seen that if you are going for a bookie automata label modeling in the states basement states base level, then the model is has be so large that leveling or model checking becomes a infusible job.

So, in this 4 and 5 lecture series we are going to the something call a symbolic model checking, that is what we are going to do in this case.

(Refer Slide Time: 04:21)

BDD and Sequential Circuits

- We can represent all Boolean functions by ROBDDs.
- Combinational circuits can directly be expressed as a Boolean expression and so we can obtain an ROBDD for them.
- In combinational circuit, the output depends on the input of the circuit.
- In sequential circuits, the outputs depend on current input and the previous outputs (states). So the Boolean expressions representing sequential circuits have two copies of output variables, one representing the current output and the other copy representing the previous output.



In this case basically, we will try to use the concept of BDD and we will try to model sequential circuits or because you know that bookie automata is nothing, but some kind of a some kind of a state machines with atomic propositions labeled in the states etcetera. But again the main problem of CTL or LTL model checking was that the bookie automata has it is repairs on explicit state enumeration, the model itself become so large

that labeling itself become a very difficult problem, because of the size of the states space and we cannot go beyond a large circuit.

So, what we are going to do in symbolic model checking, we will not explicitly design the final state machine or the state machine corresponding to the bookie automata; rather we will try to represent the whole automata in terms of binary decision diagram. So, of course it will be a very very complicated data structure and then we will try to go for the labeling algorithms or rather the model checking on the BDD itself rather than enumerating them in the states space model that is the bookie automata again do sum up.

What we will do bookie automata very large very difficult to do model checking. So, we will represent bookie automata in terms of binary decision diagram once you are able to do that, then you have to try to find out how modeling algorithms can be applied to such BDDs; basically the idea is algorithms are very well developed that same algorithms which we are seen for labeling the states like they exist for all in all parts in every state.

So, all those formulas model checking for temporary formulas use for model checking, we see that we label the different states. So, in case of high level decision diagram arithmetic decision diagram these labeling algorithms are still in the development phase, but fortunately if you go for a bit level representation and then converting it into binary decision diagram that is representation of the bookie automata or state machines in terms of binary decision diagram.

Fortunately those labeling algorithms can be very easily adopted in terms of BDDs that is actually called symbolic model checking, because we not explicitly model the states whether we model then in terms of BDD and then will go for the labeling algorithms that is why the terms called symbolic and fortunately those labeling algorithms are model checking can be very easily adapted to BDD structure, that is what again is the very beautiful of BDDs.

But if you talk about high level decision diagrams or arithmetic decision diagrams those labeling algorithms are not very straight forward and still people are doing were to find out; how they can be adopted. Anyway let us come down to the top, means lecture which are going to see today basically with what two things we have to see first thing, because already we have seen BDD in a very simple form that is there is the binary is a Boolean

function, how it can be represented in terms of BDD let me see if these a combinational circuits how can we design it.

Now, basically now we see how to adopt this whole thing, so that we can model final state machines that is the first thing we are going to learn, then next where going to see is that how the labeling model checking algorithms can be applied on those BDDs, so that you can do model checking in the BDD version other than the explicit model representation on the bookie automata that is what we are going to see.

So, we know that all we can represent Boolean function by ROBDDs is a combinational circuit. So, that is why combinational circuits and BDDs are about direct correlation, but there does not actually directly apply for the sequential circuit because, in combination of circuit the output depends on the on the input of the circuit. So, it just a combinational cloud input and output very easily you can make the BDDs.

So, combinational circuits directly correlation to the BDD, but in sequential circuits slightly allow the philosophy because, the output of the sequential circuit depends on two things is the input as well as the present state and again the output that is there is basically two types of output in sequential circuit, one is the next step and one is the primary outputs. So, basically you have to capture these two things.

So, output will be depending on the present state as well as the primary inputs. So, that you can thing in a combinational cloud like in something happen like this if this is the register, there will be a combinational cloud this will be the your input and basically and again some output from this and there will be another combinational cloud this will be your output and this is your basically next state.

So, this part is basically you can thing as so the other part you can thing as a combinational cloud that is the output, it will depend on the primary inputs basically and as well as the output of this present state that is the present state and because of the combinational cloud will make the primary output. So, these are combinational cloud directly you can use it in terms of BDDs, but only one thing is that you have to just thing that this another set of output which is over here, then decides on the next states that is in the register.

So, this again a combinational cloud, so these 2 parts you have to model using BDDs. So, that you can model the and tell sequential circuit and also you have to also take care of the fact that the sequential circuit move from state 1 to state 3 and so for those for that sequential behavior has to be modeled.

So, basically in sequential circuit you have to model 2 things the states and secondly the transitions. So, both of them has been modeling in terms of BDDs then your basically job is done. So, that is we are going to see is combinational circuit direct map in sequential circuit is 2 level first you have model this states and then you have to model the transition in terms of BDDs then your job is going to done do be done.

So, sequential circuits slightly non trivial I should not call a non trivial, slightly non straight forward and we are going to first see then how sequential circuits can be model in terms of BDDs, whether ROBDDs and once it is done there will see the we have done our basic background of modeling the sequential circuits or sequential systems which are in terms of states space that was the keeling factor in verification because, the system because very large because the verification complexities exponential in the number of state variables and it is not exponential in the terms of the size of the formula.

Because for each part of the temporary formula to be verify you just have to go for labeling. So, it is just the graph traverse or some of if you can reduce the graph size or the model size then our basically job is done. So, that is what we are going to do first we will model sequential circuits within BDD. So, once that is done your background job is done now on that BDD will try for go for this label labeling algorithms, so will go step by step. So, basically this is the slide is I have told you, so sequential circuits are express in terms of state transition diagrams, so we have some inputs and some outputs and some variables.

(Refer Slide Time: 10:16)

BDD and Sequential Circuits

- Sequential circuits are expressed with the help of state transition diagrams.
- In the case of sequential circuits, we have some inputs, some outputs and some state variables.
- The state variables can either represent current states or next states. The next state depends on current state and input variables.
- Thus, the next states can be represented by OBDDs.
- Even the transitions can be represented by OBDDs.
- Thus, we can represent all sequential circuits with the help of OBDDs.

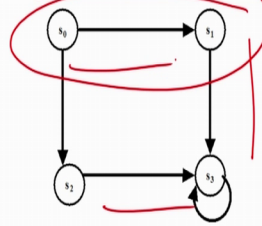
So, apart from if you compare to combinational circuit sequential circuits are inputs outputs and some state variables. So, this is the additional things which you have to model, the state variables can be either they the state variables either represent the current state or the next states. So, there are two kind of variables input that is the next state and the present state, the next state depends on the current state and input variables and the output depends on the present state as well as the input.

So, these are very well known digital design fundamental which we can do it. So, again these step is very very important the next step you have to represent using BDDs and the transitions also we have to representing terms of BDDs; therefore, this states as well as the transition they have to be representing in terms by BDDs then your job is done.

So, in combinational circuit it is just one step here will be two steps as we are going to see so and then basically with this philosophy we will see then how we can model sequential circuits or state machines also using BDDs that is what is the idea. But the basic philosophy is written in the slide which I am telling you overly is depicted in the slide the basically we have to model this states and you have to model the transitions.

(Refer Slide Time: 11:21)

Representing subsets of sets of states using OBDDs



Here, the states s_0, s_1, s_2 and s_3 can be distinguished using two state variables, say x_1 and x_2 . Let us represent them as follows.

$$\{s_0\} = \overline{x_1 x_2}$$

$$\{s_1\} = \overline{x_1} x_2$$

$$\{s_2\} = x_1 \overline{x_2}$$

$$\{s_3\} = x_1 x_2$$

So, a simple figure I am showing you so will just try to motivate this. So, this is your simple state machine you can see the transitions. So, there are 4 states, so 2 variables will be there. So, let us first enumerate them so basically this is enumeration S0 S1 S2 and S3, so you can see S0 is 00 S1 is 01 S2 is 10 and S3 is 11. So, this is very straight forward now what is we are going to do here is that we make super set of this thing because, as what the philosophy will come as I told you that basically the exponential number of states are numerical to represent any model only the require states are use to model this circuit.

So, what will do all are system, so basically we first go for a supersets kind of a representation that is tell you what is there what do I mean about that is something like this.

(Refer Slide Time: 12:04)

Representing subsets of sets of states using OBDDs

Since we use Boolean functions to represent subsets, we can represent all the subsets of states (s_0, s_1, s_2 , and s_3) using OBDDs.

$$\{s_0, s_1\} = \overline{x_1}x_2 + x_1\overline{x_2}$$

$$\{s_0, s_2\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_0, s_3\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_1, s_2\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_1, s_3\} = \overline{x_1}x_2 + x_1x_3$$

$$\{s_2, s_3\} = \overline{x_1}x_2 + x_1x_3$$

$$\{s_0, s_1, s_2\} = \overline{x_1}x_2 + x_1x_2 + x_1x_2$$

$$\{s_0, s_1, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_3$$

$$\{s_0, s_2, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_3$$

$$\{s_0, s_1, s_2, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_2 + x_1x_3$$



The state S0 or S1 if all is this two states are their this is the 0 0 or 0 1, similarly S0 and S2 is 0 0 or 1 0 and last 1 will be your full set that is equal to 1 0 0 0 1 1 0 and 1 1. So, basically I am enumerating all the power set of the set of states that is all possible combinations, so instead I have not shown S0 S1 and S2 and S3 that is obvious basically this is the 4 states and then they are the all the combinations.

So, what will do basically will first make the superset of the set of states. So, this combination will here it is depicted to show, but basically you have to pick up only those states which are elements for the design and then we will try to represent them using BDDs and the transitions.

So, first what is the goal you have to pick up only those states which are relevant for our discussion, one thing you have to understand that also in case of a system design we draw only the relevant states, but when you are trying to represent and final state machine in actually blows up because it is not a compress diagram. Here what is happening here also we are being the same thing that only the relevant states we are bringing into picture, but will be represent them into BDDs through BDDs.

So, all the redundant elimination redundant states which are not there basically redundant structure, that is the states which are not represent in the systems will be eliminated out in the BDD representation, that is the certain difference that if you are going for a simple final state machine representation even if the number of states are limited, because you draw only the practical states or which are the reachable states or in fact basically which

are the states under consideration, but if you are using a normal algorithms which is not in a form of BDD even the states which are not relevant also comes into the enumeration process, because it is a simple state present enumeration like a binary decision tree.

But if we represent in terms of BDDs we will represent only the relevant stresses to us and the other parts which are irrelevant because, they are neither reachable they do know do not there are not matter in the computation. So, it will be happening something like this as we have already seen that if both the paths lead to 0 that is both variable X if 0 and 1 both has the same value as the output then this state this eliminated.

So, whatever states is not relevant that it not model explicitly basically get eliminated in this manner, as told you that that is why the idea BDD will take a subset of this state as you have see like for example, if only S0 and S1 are in equivalent to that we will represent this Boolean function as a in terms of BDD, so it will be a very complex data structure.

(Refer Slide Time: 14:36)

Representing subsets of sets of states using OBDDs

Since we use Boolean functions to represent subsets, we can represent all the subsets of states (s_0, s_1, s_2 and s_3) using OBDDs.

$$\{s_0, s_1\} = \overline{x_1}x_2 + x_1\overline{x_2}$$

$$\{s_0, s_2\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_0, s_3\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_1, s_2\} = \overline{x_1}x_2 + x_1x_2$$

$$\{s_1, s_3\} = \overline{x_1}x_2 + x_1x_3$$

$$\{s_2, s_3\} = \overline{x_1}x_2 + x_1x_3$$

$$\{s_0, s_1, s_2\} = \overline{x_1}x_2 + x_1x_2 + x_1x_2$$

$$\{s_0, s_1, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_3$$

$$\{s_0, s_2, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_3$$

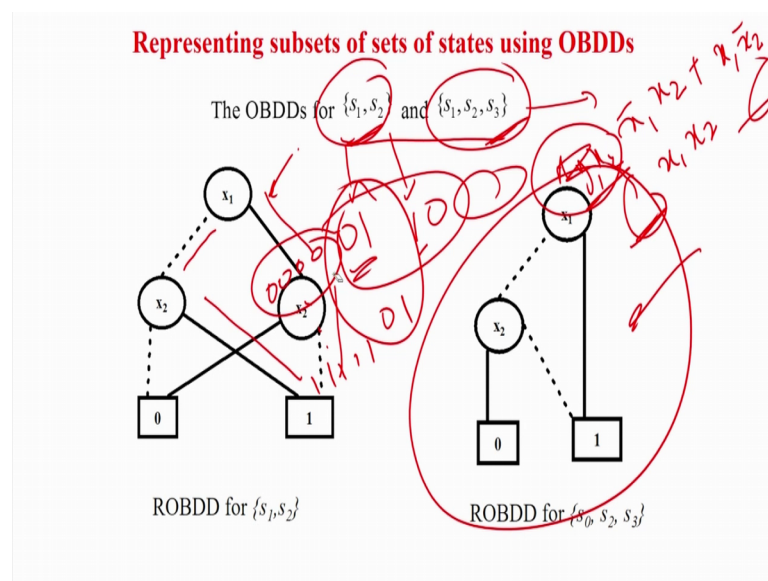
$$\{s_0, s_1, s_2, s_3\} = \overline{x_1}x_2 + x_1x_2 + x_1x_2 + x_1x_3$$

But if we go for a explicit state enumeration, so we will have only S0 and S1 say. So, it is actually something called 0 0 and this is 0 1, but the other states which are non relevant like 1 1 0 and 1 1 we still the represented in the all in the algorithm. If will trying for state for some sort of all sort of algorithms these two is also try to put their presence, as it happens in case of a binary decision tree, because in binary decision tree if you the variable x 0 may be is a 0 and this side may also same is also 0, but we will have to did 2

explicitly if knows because $x = 0$ is also 0 $x = 1$ is also 0 . But you still explicitly enumerate actually x is a redundant variable would not require that if you go for a simple state modeling first things will try to appear, but if you are going for a BDD and based on representation then those redundancies will be gone.

So, what is the first job all these power sets of the states are done then will take some relevant states set which is require for the modeling and only that will be the represent in terms of BDD, so will get a very Complex structure right.

(Refer Slide Time: 15:38)



For example, if you see S_1 and S_2 and $S_1 S_2$ and S_3 , so say for example for certain reason these are the states such which are relevant. So, S_1 is 1 so S_1 is nothing S_1 and S_1 is 0 S_1 is 1 S_1 is basically nothing, but S_1 and S_2 , so this is 0 1 and this is 1 0. So, this is basically represented by this BDD. So, if you see x_1 is equal to 0 and $x_1 x_2$ is equal to 1 your landing it to 1, so it is actually representing this state.

Similarly, x_1 equal to 1 and x_2 equal to 0 basically this enumerating this, so this are the two transitions which are the lines I have drawn represent the two; basically this two transitions is represented by the BDD. Similarly if you are considering S_1 , S_2 and S_3 you are going to get this BDD structure because S_1 corresponds to 0 1, S_2 corresponds to 1 0. I have drawn the paths and showed that they are leading to 0. So, basically if you say S_1 , S_2 and S_3 are BDD seen so small and but if you see S_1 , S_2 and S_3 anyway such S_1

is what S1 anyway all the enumerations I have not done basically, it will be it will 2 the power 4 it will be 16.

(Refer Slide Time: 16:56)

Representing subsets of sets of states using OBDDs

Since we use Boolean functions to represent subsets, we can represent all the subsets of states (s_0, s_1, s_2 and s_3) using OBDDs.

$$\{s_0, s_1\} = \overline{x_1} \overline{x_2} + \overline{x_1} x_2$$

$$\{s_0, s_2\} = \overline{x_1} x_2 + x_1 \overline{x_2}$$

$$\{s_0, s_3\} = \overline{x_1} \overline{x_2} + x_1 x_2$$

$$\{s_1, s_2\} = \overline{x_1} x_2 + x_1 \overline{x_2}$$

$$\{s_1, s_3\} = \overline{x_1} x_2 + x_1 x_3$$

$$\{s_2, s_3\} = \overline{x_1} \overline{x_2} + x_1 x_3$$

$$\{s_0, s_1, s_2\} = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 + x_1 \overline{x_2}$$

$$\{s_0, s_1, s_3\} = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 + x_1 x_3$$

$$\{s_0, s_2, s_3\} = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 + x_1 x_3$$

$$\{s_0, s_1, s_2, s_3\} = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 + x_1 \overline{x_2} + x_1 x_3$$

24 - 16
= 8

So, this I have not drawn but obviously, you can find out that if you will have 3 terms because, S0 will be equal to $x_1 \bar{x}_2 \bar{x}_3$ this 1 will be basically $x_1 \bar{x}_2$ plus $x_1 \bar{x}_2 x_3$ this is for then $x_1 x_2$ huh. So, this was going to be the expression for this, so you can see there is the size is quite large corresponding to the BDD representation and even the thing of not only for variables if the system is very large basically you will also have say $x_1 x_2 x_3$ 1 $x_1 x_2 x_3$. So, many number of elements will be added over there if you think many number of variables are very large. So, the expression will be very large if you are just consider a 3 stage, but actually all other variables is present are basically redundant. So, BDD representations we eliminate all this and you are going to get a very compress structure.

So, what we what do we have learn till now the basically will have this state set which is relevant for us and we represent using BDDs. The advantage will be all the redundancies variable, which are not relevant to represent the present states will be eliminated and we are get a going to get a very compress representation of the state set these only a 4 2 variable. So, this is not obvious and as I told you if you 100 variable set and we are only in interested in these two states may be the all other additional variables like which will

have 0 0 0 all combinations, basically say 0 0 0 0 0 through dot dot dot 11 1 1 1 this is 0 1 this will remain same all will be embedding in the state enumeration representation.

But we are redundant because the cancel out and whenever you go for a BDD base representation we get a very small diagram basically. So, that is why the first job is we represent the relevant states has using BDD and we get a compress structure.

(Refer Slide Time: 18:52)

Representing transitions using OBDDs

- For representing transitions, we need to use more variables. For example, if n variables were initially used to represent the states, now we need to have n more variables.
- Suppose, initially we used variables $x_1, x_2, x_3, x_4, \dots, x_n$ to represent the states, now let us introduce more variables $x'_1, x'_2, x'_3, x'_4, \dots, x'_n$, so that we are able to represent the state transitions.
- $x_1, x_2, x_3, x_4, \dots, x_n$ are used to represent the current state of the transition and $x'_1, x'_2, x'_3, x'_4, \dots, x'_n$ are used to represent the next state of the transition.

Now, we have to represent the other part which is basically nothing but you transitions. So, what you have to go for transitions basically transitions, if these are your variables we have to have another set of variables by the primes.

Now, why do you want to do it because, basically x this non prime value will represent the present state and the this 1 will be represent the basically represent the next state, because as I as I told you that we have three step in case of consequential circuits inputs present state output and sorry present state, next state, inputs and 1 in the primary output. So, primary output as now we are not dealing with directly, because primary output is a combinational circuit we have already shown.

So, inherently basically we can say the we have inputs present state and next state, output also you can thing as a element for the time being you are not making the things more complicated. So, therefore this 1 will all the state variables will be representing the

present state and the prime versions will be representing the next state. So, we have to just duplicate the state variable size now will see how to do this.

Now, you have to how do you represent the transitions. So, I will just very theory let us say you are looking at this. So, basically if we see S0 to S1 is a transition, if you look I mean just give you 1 example it will be clear. So, you can see from S0 to S1 is a transition how do you model it, just rename the S0 to S1, S1 to S3, S2 to 1 2 3 4 and some 5 transitions are there we have the enumerate each of the transitions first.

So, let us take the examples of S0 to S1, so how do you do it let me show it for you. So, here you have seen the variables are x 1, x 2 and x 1 prime and x 2 prime, so basically x prime are the next state. So, S0 means x 1 bar x 2 bar they are the present state, what is the next state next state is S1 that is equal to 0 1, that is x prime bar x 2 bar that is basically nothing, but is goes from 0 0 to 0 1.

(Refer Slide Time: 20:56)

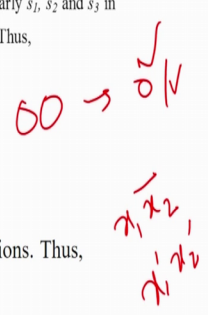
Representing transitions using OBDDs

If s_0 occurs as a next state, then we use $\overline{x_1}x_2'$ to represent that. Similarly s_1 , s_2 and s_3 in next states are represented by $\overline{x_1}x_2'$, $x_1\overline{x_2}'$ and x_1x_2' , respectively. Thus,

$s_0 \rightarrow s_1$ will be represented by $\overline{x_1}x_2x_1'x_2'$
 $s_2 \rightarrow s_3$ will be represented by $x_1x_2x_1'x_2'$
 $s_0 \rightarrow s_3$ will be represented by $\overline{x_1}x_2x_1'x_2'$
 $s_3 \rightarrow s_1$ will be represented by $x_1x_2x_1'x_2'$

Thus, we can represent all transitions by Boolean functions. Thus, we can construct OBDDs for transitions also.

A state transition graph (STG), which is used to represent sequential circuits is just a subset of all possible transitions.



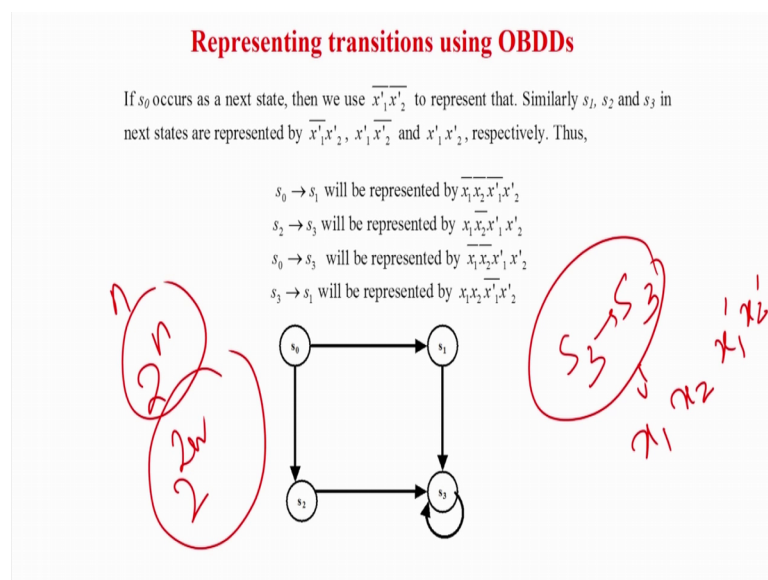
So, that is this is equal x 1 prime, x 2 prime and this is x 2, x 1 prime bar and this is x 2 prime. So, that is what is represented by this formula.

Similarly, x 2 to x 3 is what x 2 is basically we here, so x 2 is nothing but x 1 x 2 bar and where it is going x 3 that is x 1 bar x 2 bar that is 1, so you are representing this by this transition. So, whatever transition is very simple this 1 will corresponding to the state encoding for the present state, the this state will be state encoding for the next state

another we have to have a prime version. So, this way S0 to S1, S2, S3, S0, S3, S1 is the whatever transition you will have you have to actually enumerate in this form and then basically you have to do an because, all the first transitions are present and represent using a BDD your job is basically done.

So, that is what is the idea, so you take a set of all the transitions present state distraction state present state distraction state, you represent in terms of the binary variables or the Boolean state variables present state will be non prime next state will be prime and you make a or representation and that is going to be the BDD representation, so just you can see you can have.

(Refer Slide Time: 21:56)



Just given this slide you can just see over here then how transitions are the represented.

So, I think we have the have S3 to S3. So, at will be equal to x 1 bar x sorry x 1, x 2 and x 1 bar, x 2 bar. So, this is will be representing this transition, so this where that is the so very simple. So, just you have to enumerate all this things and then basically you have to represent.

(Refer Slide Time: 22:22)

Representing transitions using OBDDs

- In the example under consideration, where there were 4 states s_0, s_1, s_2 and s_3 , 16 possible transitions may be there.
- The STG will contain only some of those transitions.

The STG of the example, consists of the transitions

$$\{s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_3, s_2 \rightarrow s_3, s_3 \rightarrow s_3\}.$$

So, the STG can be represented by Boolean expression:

$$f = \overline{x_1} \overline{x_2} \overline{x_1'} \overline{x_2'} + \overline{x_1} \overline{x_2} \overline{x_1'} x_2' + \overline{x_1} x_2 x_1' \overline{x_2'} + \overline{x_1} x_2 x_1' x_2' + x_1 x_2 x_1' \overline{x_2'} + x_1 x_2 x_1' x_2'.$$

S3

So, what will going to happen is they were 5 transitions 1 2 3 4 5, 5 transitions are there. So, basically the 5 trans will be there 1 2 3 4 and 5, so this 1 this 1 if you look at it corresponds to S3; S3 that is the self loop similarly all other terms can be found out.

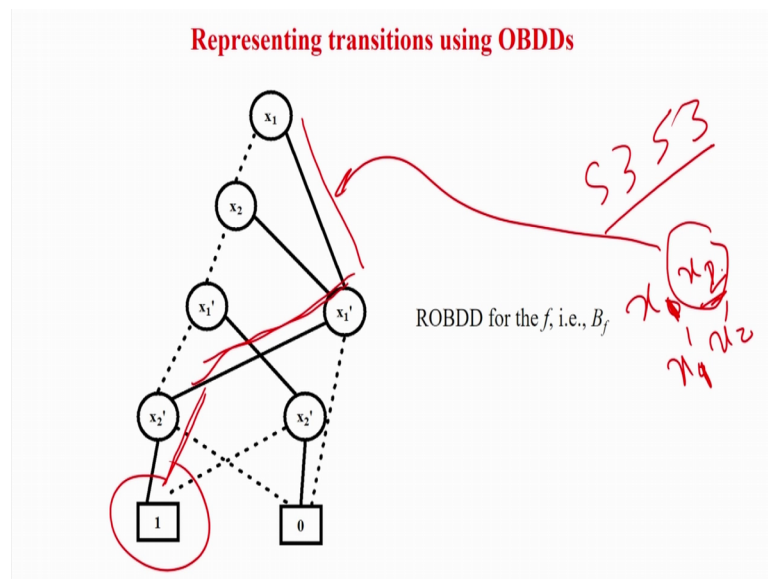
So, basically this is your Boolean function which corresponds to basically you are transitions. So, if again it is written that as there are 4 states. So, it is a fully connected graph there will be 16 possible transitions. So, if you are going for a flat representation in the binary decision tree philosophy it will be again exponentially blowing up; but we all know that all graph criterial graphs are very very passed.

So, in this case I have taken a practical example sorry a dummy examples, so you have at least 5 transitions. But in the practical systems where you have very large states does the number of transitions are much much smaller than 2 to the power or n square where n is the number of states, basically if there n variables 2 to the power n is all possible states and the number of all possible transitions are 2 to the power 2 n that is the square.

But in real life you will find out that this number of states and number of transitions are much much lower than basically the number of correspond to 2 to the power n; that means, if there 100 variables number of states will be very much lower compare 2 to the power 100 and the number of transitions will also be drastically lower than 2 to the power 200, basically that is why in BDD; we model only this important transitions or the transitions on the consideration and if you are using a binary tree to represent it.

Even if the unnecessary information will also going and it will things makes the things blow up. But in binary decision diagram representing will cramp up eliminate the redundancies that is it will mainly model only this trans that also, if there is a redundancy among them there will be eliminated and then will have very compress representation of the transitions. So, now what we have?

(Refer Slide Time: 24:12)

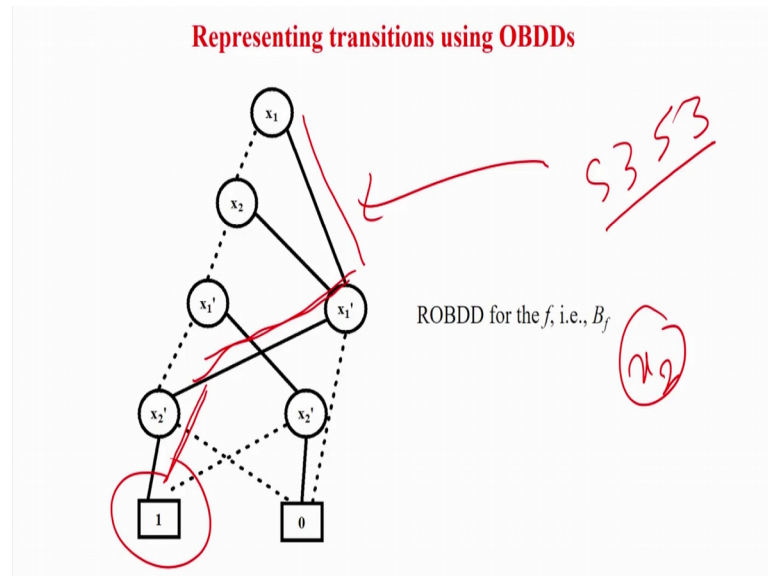


We this is basically you are the, this the BDD, if you look it is the BDD corresponding to this transition let us trace 1 transition. So, x_1 equal to 1 and if you see x_1 bar equal to 1 and this is 1 transition let me I am talking about S3. So, what is S3? S3 is $x_1 = 0, x_2 = 1, x_1 = 1, x_2 = 1$ bar x_2 bar, so that transition should be to 1. So, if you see x_1 equals to 1 then x_1 bar equal to 1 and x_2 bar equal to 1 you are going to a 1. In fact, you see here once you from this compression still you are saving on 1 variable, but is you are saving on x_1, x_1 bar x_2 bar, so you are saving on this variable. So, will this part actually corresponds to these transitions 1. Basically, this x_1 bar is sorry which is saved x_1 sorry this x_1 is saved sorry not this 1 we are saving actually x_1, x_1 bar x_2 bar.

So, x_1 is there x_2 this is the variable we are actually saving to represent this transition, so x_1, x_1 bar x_2 bar the answer is a 1. In fact, we do not require x_2 to explicitly represent this transition. So, even after just instead of 2 to the power 16 possible transitions, we can just represent by this function using BDD and in that also if you compare the number of paths, because the number of variables are 4 into 4 4 4 4 16 5

sorry 1 2 3 4 5 was a 20, but we will not 5 20 transitions in this binary decisions diagram. So, not only the 16 possible transitions even each of this explicit terms in this expression are also not explicitly require to model in the BDD, because again we can go for a cramp representation. So, these transitions basically $S_0 S_2 S_3$ bar is represented by this and here also the term basically x_2 is eliminated.

(Refer Slide Time: 25:58)



So, you can see basically we have a very compress represent for the state subset as well as we have very compress represent as for the transactions. So, now we have represented the model of any system is terms of state machines, because everything has memory because you cannot have any system practically designed without having states. So, we have represent states using BDDs how by modeling the state subset and modeling the transition in a very cramp manner using BDD.

So, this brings us to the half way that we have given a very nice platform, because the again the BDD that is why I told you one of the most beautiful data structures I have personally seen in my life is binary decision diagram; whole with VLSI cad industry standing on this data structure people have again move array from this that is true, but that is what is the founding stone.

So, basically we have seen, but this again this BDD is coming to a boon, so represent state machines states and transitions in a very compressed manner in a system is represented and we have already discussed that only this state modeling is problem, once

this state is modeled this model checking that is your labeling algorithm is not that complex.

But now we will see that unlike for high level decision diagrams etc, labeling is very very straight forward using a binary decision diagram that is what we are going to see now that is actually called a symbolic model checking. Why it is name is symbolic because we are not going to expressively represent the bookie automata or the state machine in terms of FSN kind of a structure. Basically, we are going to use BDD representation and we will go for the labeling so this is quite long. So, we will today we will do half of it and then in next class we will try to basically complete it.

(Refer Slide Time: 27:34)

Symbolic Model Checking

- CTL model checking algorithm.
- It is a labelling algorithm and we label the states by the given CTL formula if the CTL formula holds in the state.
- The complexity of the algorithm is related to the size of the Kripke structure.
- For complex system, the size of the FSM is prohibitively big and so the model checking takes huge amount of time.
- Finite state models of concurrent systems grow exponentially as the number of components of the system increases.
- For example, an FSM having 'n' state variables will have ' $O(2^n)$ ' finite states, i.e., the relation between number of state variables and number of finite states is exponential.

So, we have seen that the CTL is a CTL model checking algorithm or LTL model checking algorithm already seen basically, it is nothing but a labeling algorithm. So, if the graph is very big there is the input space is input graph is very big you are going to kill it, because this states place is basically kill it.

Now, what do you have to done because the complexity of the algorithm is related to be Kripke structure basically this is nothing, but your system model. So, for complex system the FSM is prohibitively big you have already seen that if you are n variables it become 2 to the power n and if you are going to keep binary decision tree kind of a representation that is the flat representation everything is going to blow up, it will 2 to

the power n and number of basically your transition can be order of 2 to the power $2n$ that is the square of it and everything is going to die now.

(Refer Slide Time: 28:15)

Symbolic Model Checking

- It is known widely that the state space explosion problem in automatic verification has limited its use to small systems.
- To make the model checking more effective, we need some methods to content the state space explosion problem.
- One possible way is to represent the state space of the FSM in a compact way so that the model checking becomes effective
- With the help of ROBDD we can represent the Boolean expression in canonical form. Also we have seen that ROBDD can be used to represent the Finite State Transition system and the set of states of transition system.
- In model checking algorithm we use ROBDD to represent the Kripke structure and the method that we get is named as Symbolic Model Checking. It says that the states are not enumerated but represented symbolically as a set with the help of ROBDDs

EX
AF

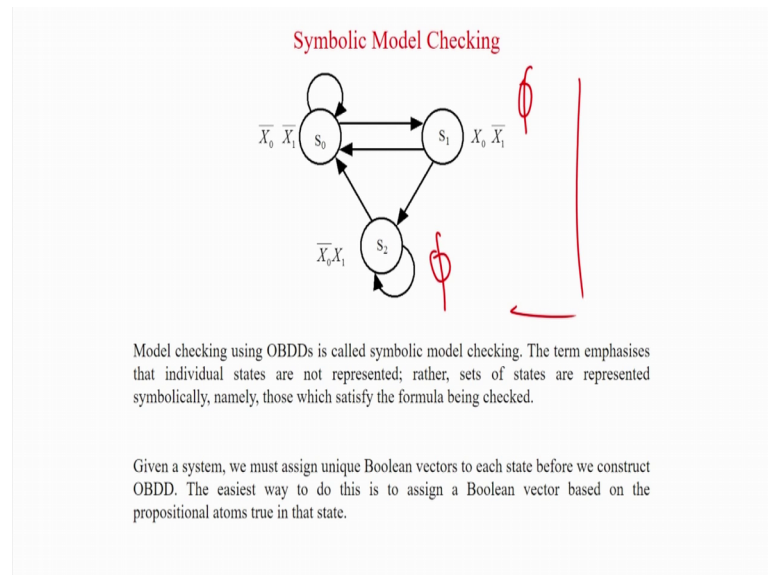
So, basically what so that is why basically this state space exp explosion problem in automatic verification, basically we have still going at a very low level circuits or low level systems. Truly speaking easily if the BDD we have still not actually able to reach this so called code uncode the noc verification, soc verification as an entire block basically because as I told you still the BDD is a bit level and the binary decision diagram at the higher level like HLBDD ZBDD etcetera are there, but still the labeling algorithms are not been well developed in those cases.

So, still with BDD we can take we have taken the verification to a much much higher level, but still where you are going for system level verification like nocs and socs we had being in parts and persons like we will may be verify the router we will may be verifying 1 core. But it is very difficult to verify the entire noc or soc taking it is details into picture. So, that is the very very important research challenge and many people are working towards it, so anyway let us come back to our problem again.

So, what you are basically doing we are representing the FSM in a very compact manner using for ROBDDs. So, that the representation of the Kripke structure has become very efficient and now what we are trying to do we will try to basically develop all the labeling algorithms like EX all path in future etcetera, all those how can we do basically

on the BDD representation rather than on the state space representation. So, as this is done symbolically, so the name call symbolic model checking, so now we are going to dive into that again let us take a simple structure.

(Refer Slide Time: 29:47)



So, basically explicitly we have to show this structure because, otherwise we will not be able to appreciate the fact; but again from the previous experience nobody's again find the first going to make this a finite state machine and then going to make the BDD that nobody's going to do from the system we directly make a BDD representation. But for elastration basically we are going to have first prepare structure like this and then basically very one important because, first I have told that first we have a BDD structure to represent a sub set of state.

Now, why is it because we know that in model checking we may say that some atomic proposition say high is true only at a state and maybe it is true only at a state, how do you represent it? That is by representing a sub set of state that is why the first goal is to represent a sub set of states. So, in this case in the example they are saying whatever is the atomic proposition is basically say that it is true say S 1 or whatever sorry x 1.

(Refer Slide Time: 30:35)

Symbolic Model Checking

Model checking using OBDDs is called symbolic model checking. The term emphasises that individual states are not represented; rather, sets of states are represented symbolically, namely, those which satisfy the formula being checked.

Given a system, we must assign unique Boolean vectors to each state before we construct OBDD. The easiest way to do this is to assign a Boolean vector based on the propositional atoms true in that state.

Some atomic proposition called x_1 whatever it is rainy or heterozoan some proposition which is true only a state that is what is the assumption basically right. So, this is the structure keep it in mind and we are assuming that some atomic proposition which we are taking into picture is says the x_1 , is true only in state x_2 .

(Refer Slide Time: 30:53)

Symbolic Model Checking

Model checking using OBDDs is called symbolic model checking. The term emphasises that individual states are not represented; rather, sets of states are represented symbolically, namely, those which satisfy the formula being checked.

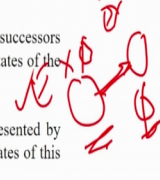
Given a system, we must assign unique Boolean vectors to each state before we construct OBDD. The easiest way to do this is to assign a Boolean vector based on the propositional atoms true in that state.

Now, we will try to go for the model checking on labeling in this structure and we will today we will see only for one operation and like EX today, we are going to study only for EX another operations will see in the next class, because this is actually it is a quite long to visualize how it happens.

(Refer Slide Time: 31:11)

Symbolic Model Checking

- Generally, in model checking algorithm, the Kripke structure M and a CTL formula Φ are given as input and the output of the algorithm is the set of states of the model which satisfy Φ .
- While checking for Φ , we know that the model must be labelled by the sub-formulas of Φ . Also it is clear that the primitive sub-formulas are the atomic propositions.
- From the labelling function of the Kripke structure, we know the states in which an atomic proposition p is true. This set of states can also be represented by an OBDD.
- The basis of model checking algorithm is to find out the previous state of a given state or a set of states.
- Consider the labelling algorithm of $EX \Phi$. Label any state with $EX \Phi$ if one of its successors is labelled with Φ . While going to label the states with $EX \Phi$, we must label the states of the Kripke structure where Φ is true.
- Therefore, we know the states where Φ is true, and this set of states can be represented by OBDD. The job of model checking algorithm for EX is to find the predecessor states of this state.



So, what basically happens so in a model checking algorithm we have a Kripke structure and a CTL formula, so now the problem is with the complexity of m . So, now we have we bring it down by BDD representation. So, what we do we know that for checking this we have to model the sub or label the sub formulas of ϕ in the states. So, basically that means, what for initially we will have some states where the formula that is or sub formula is true at some of these states, that is the first state at some of the state this is going to be true.

So, in BDD representation first we are going to represent those states as a sub set, already we have seen how to represent the subset in BDDs. So, we will represent as the subset then we will be then from the labeling function of the Kripke structure we know the states in which is the atomic proposition is true; that means, first we have to label the states which are directly which are having this as true that is what we are doing, so these states can be represent as the subset of sub this states subset is represent the BDD.

So, what is say the first statements is that we are having m in the Kripke structure in BDD form and formula ϕ then what we do ϕ is then ϕ is broken down into atomic propositions and then we will find out in which states the ϕ is true that will actually that subset will be representing as a BDD.

Then basically we have to find after that things will change for different formula, if it is EX some algorithm will be there EX some other algorithm is will be there EU some

other algorithm will be there, but first state we to find out the set of states I will representing BDD for the phi is true. Then basically today we will see about EX. So, the basically what we do in EX basically if there is a state and there is a state where phi is true this is going to have value of EX phi is going to be true over here, because all such parent states has to be marked with EX phi if in the adjacent state the phi variable holds true that we arrive already you have seen it.

So, what we are basically going to do first we are going enumerate all those states which are having the value of proposition phi true, that subset of states will be represented by BDD. Then we will find out an algorithm in terms of BDD which we find out these states from where there is a transition throughout the states with phi is going be true, that is what is the return over this one.

Now, we will elaborately see that algorithm this is what as I have told you this is written in the slide, so you can read of this anyway the notes will also be uploaded, so we are going to this one.

(Refer Slide Time: 33:30)

Symbolic Model Checking: EX

$\text{Pre}_\exists(X)$: takes a subset X of states (of OBDD) and returns the set of states which can make a transition into X.

$\text{Pre}_\forall(X)$: takes a subset X of states (of OBDD) and returns the set of states which can make a transition *only* into X.

How to Find $\text{Pre}_\exists(X)$:

Given,

B_X : OBDD for set of all states where X is true.

B_\rightarrow : OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD B_X' .
- Compute the OBDD for $\text{exists}(\vec{X}', \text{apply}(*, B_\rightarrow, B_X'))$ using the **apply** and **exists** algorithms.

So, one very important function which we are going talk to these pre of X that is actually the representation, therefore that means basically this is a set of states and we are going to find out this is a this is a X are set of state with some special property in this case, these are the steps where phi is going to be true and then basically pre of X means all the

So, this one actually represents states from where there is a transition to a state X with some property holds and is there exists, so there exists one transition is ϕ , but when you are going to for all operation the same thing. But all the transition from first states to good should hold for the states with this property ϕ is true, this is for all and the first one is for X ; that means, if all transitions need to states where ϕ is true then this is going to be true; that means, this is first subsets will be found out and if you recall an existential property then other pass may go anywhere I am not bothered, but there should be at least one transition for state X where the ϕ holds, so anyway first we are going to see about it.

(Refer Slide Time: 34:42)

If you see what we are trying to do we are trying to find out all those states, we are going to find out BDDs in terms of BDDs state B_x . So, B_x is the subset of states where X is true, that is in our case may be the proposition ϕ is true because, we when I given state may be only two and a two states are there where the proposition ϕ is true.

Those two state subset will be represent in terms of BDD, then we all should also have been had the transition relation or the transition level BDD or the BDD represent in the transitions; then what we are going to do then basically rename the variables of B_x to their primed versions. So, there is one BDD basically that this BDD B_x , which corresponds to all the states where the ϕ is true. Then we will actually just label the so say let us this let this is one state where the ϕ is true that is say represented by B_x

Now, we will makes B_x prime, so what do you mean by B_x prime; that means, it will virtually make this X prime, that means it will virtually make something like this that is the next state where ϕ is true, B_x means present state where ϕ is true and if I make B_x prime then it will make a dummy structure like this next states where ϕ is true, then we are going to make this operation called apply, apply dot transition into B_x . That means what we are going to do in the next state, so we have the B_x bar means something like this where ϕ is true.

(Refer Slide Time: 36:10)

Symbolic Model Checking: EX

$\text{Pre}_2(X)$: takes a subset X of states (of OBDD) and returns the set of states which can make a transition into X .

$\text{Pre}_1(X)$: takes a subset X of states (of OBDD) and returns the set of states which can make a transition *only* into X .

How to Find $\text{Pre}_3(X)$:

Given,

B_X : OBDD for set of all states where X is true.

B_{\rightarrow} : OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD $B_{X'}$.
- Compute the OBDD for $\exists X' (\vec{X}, \text{apply}(\cdot, B_{\rightarrow}, B_{X'}))$ using the **apply** and **exists** algorithms.

Then we are going to take the BDD of the transition structure that is called by b arrow mark, then if you are going to make a dot product of b transition into the next state where this one, you are going to have a structure like this so this is X , you are going to have a transition structure like this that this all the transitions in which the next state is X prime that is where ϕ is true. So, this operation apply dot B_x , B_x bar basically gives you all the transitions which are of this part present state there exist at least 1 transition.

So, the next state where phi is true because, you are taking the entire set of transitions doting it product that is dot product with something like this you are making a dot product means something like this. That means, we are now going to have some transitions like this 1 which will be after output of this.

(Refer Slide Time: 36:55)

Symbolic Model Checking: EX

How to Find $\text{Pre}_3(X)$:

Given,

B_X : OBDD for set of all states where X is true.

B_\rightarrow : OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD B_X' .
- Compute the OBDD for $\exists x. X'$, $\text{apply}(\bullet, B_\rightarrow, B_X')$ using the **apply** and **exists** algorithms.

The **exists** algorithm can be implemented in terms of the algorithms **apply** and **restrict**

$$\exists x. f = \text{apply}(+, \text{restrict}(0, x, B_f), \text{restrict}(1, x, B_f))$$

After that what we are going to do then basically now we are having something called X and we are going to have something called X bar where phi is true, this you are going to get after this 1 is there.

Now, what we are going to do now somehow we have to basically eliminate out because, we are going to have because something like this, so a transition is represent as you have seen if it is represent in terms of some $X_1 X_1$ may be X_2 bar something like this dot X_1 bar and X_2 bar something like this. So, we have already seen this.

Now, what we require we finally require this we have to eliminate this part has to be eliminated because, you want to find X where in next state if phi is equal to true; that means, this one basically this one this is going to give you the transitions. So, transitions in BDD as we have seen we will have all the prime variables or the nonprime variables.

Now, we have to somehow eliminate this part that is you have to eliminate this prime portion, so you should only keep this. So, basically operation called exists we will actually do it for you. Now what is the operation this is a very simple just you have to

just look at the philosophy slightly because, already we have discussed in the operations of BDD that is something called a Restrict operation.

(Refer Slide Time: 38:03)

Symbolic Model Checking: EX

How to Find $\text{Pre}_3(X)$:

Given,

B_X : OBDD for set of all states where X is true.

B_{\rightarrow} : OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD $B_{X'}$.
- Compute the OBDD for $\exists x. (X' \wedge B_{\rightarrow})$ using the **apply** and **exists** algorithms.

The **exists** algorithm can be implemented in terms of the algorithms **apply** and **restrict**

$$\exists x. f = \text{apply}(\wedge, \text{restrict}(0, x, B_f), \text{restrict}(1, x, B_f))$$

So, whenever you say restrict 1 X B_f ; that means, we are going to get the B if B I means BDD if some variable X is equal to 1 and also if you say that restrict something called 0, then you are going to get a BDD where X is equal to 0 and this is a BDD where X is restricted to 1; that means, how the BDD we look if X is equal to 1 then we can also make a BDD, X is some variable you can call it yz or whatever that BDD where y is equal to 0 that is also possible very well. So, this BDD corresponds to the fact that where X is equal to 0 that is going to the BDD look like. We can have another BDD where X is equal to 0, so we will have a BDD where X is equal to 0 we will have 1 BDD called where X is equal to 1. So, there are we are fixing some variables to either 0 and 1 and we are trying to find out how the BDD looks.

Now, if I make a or of it then what we are going to get that is actually call the add. So, you are having 1 BDD where X is equal to 0 we are having 1 BDD when X is equal to 1 and you are or it then directly from Boolean representation, find out the BDD which is irrespective of x ; that means, this 1 if X is equal to 1 if I or with X equal to 0 resultant would be something which is nondependent on x ; that means, the sub part of the BDD which has nothing to do with X equal to 0 or X equal to 1, that is independent of x .

Similarly, we can do with n number of variables that is why something called an exists operation. So, what exists operation does basically it eliminates some of the variables from that function; like for example, may be I have some function called X_1 bar and some function called a is there and then you have some function called X_1 then again a is there.

So, if you can easily find out, but here basically this X_1 is not very much required, so even if I fix it I am going to get some BDD; the idea is basically say Boolean sub Boolean function it is there we want to find out only that part of the function we does not get changed with that X is equal to 0 or X is equal to 1 ,may be I have a very big function, but only few terms will remain if X is either 0 or X is either 1.

We want to bring a those terms that is some part of the functions we will remain some for some parts of the formulas will remain true, whether X is equal to 0 or X is equal to 1. So, actually apply will bring it out right for example, I told you X equal to 1 means that part of the BDD which is true if X equal to 1, X equal to 0 means that is a restrict operation that part of the BDD which is true when X is equal to 0 or it.

So, the resultant BDD will be something which is true if X is equal to 0 or X is equal to 1, that is actually called the apply operation and apply operation basically eliminates some of the variables. So, now basically this 1 as we have seen what it does, it represent something called X something called X prime and there ϕ is equal to true and as I told you it is represented in terms of some variables like $X_1 X_2 X_1$ bar something of this terms actually it is there.

(Refer Slide Time: 41:00)

Symbolic Model Checking: EX

How to Find $\text{Pre}_\exists(X)$:

Given,

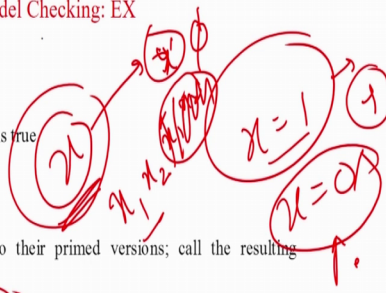
B_X : OBDD for set of all states where X is true

B_\exists : OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD $B_{X'}$.
- Compute the OBDD for $\exists x. f$ using the **apply** and **exists** algorithms.

The **exists** algorithm can be implemented in terms of the algorithms **apply** and **restrict**

$$\exists x. f = \text{apply}(+, \text{restrict}(0, x, B_f), \text{restrict}(1, x, B_f))$$


Now, we have to just retain this terms we have to eliminate this terms, so basically what we have to do we have to apply \bar{X} , here \bar{X} means basically all the terms which are in the inverted form like it $x_1 \bar{x}_2 x_3$ prime or available. So, we have to apply x_1 prime x_2 prime and x_3 prime or the parts of the BDD we have to bring it out which are irrespective of X prime equal to 1 X prime equal 0 and so for, then you are going to actually get this term out.

So, basically that is what they are doing, so \bar{X} these are nothing, but your vectors. So, I having this transitions if I apply exists; that means, from the time being you just understand that and eliminating this parts, only if only those transitions will come out where x_1 and x_1 prime are irrelevant for them either 0 or 1 you are going to get the same thing, so you are going to get this transition as a output.

(Refer Slide Time: 41:48)

Symbolic Model Checking: EX

The working of this algorithm is explained as follows.

$\vec{X} = \{x_1, x_2, x_3, \dots, x_m\}$ is the vector of present state variables.

$\vec{X}' = \{x_1', x_2', x_3', \dots, x_m'\}$ is the vector of next state variables.

- Consider a set of states X and its OBDD representation is B_X .
- The variables involved in this OBDD are from the vector \vec{X} .
- We know that if we rename the variables of a Boolean expression, the truth value of the expression remains same. Therefore, if we rename the variables of an OBDD, the resultant OBDD still represents the same Boolean expression.
- If we rename the variables of B_X to get a new OBDD $B_{X'}$, then $B_{X'}$ represents the same set of states like B_X , only changes are the variables which are from the vector \vec{X}' , that is, represented by next state variables.

B_X
 $B_{X'}$

I will give you an example that will make the things very very clear. So, basically let us see that X prime and X prime bar are nothing in this case, if you see they are nothing but your basically your state variables and prime version is the next state variable. So, first what we do consider set of state X and represent B_X , B_X is the subset of state where the formula is true we are represent this vector.

Now, basically what we do basically we have the B_X bar is the next state and then B_X is the present state. So, here we will have all the nonprime variables, here we will all have the prime variables. So, that is so this is actually they are represented by X bar.

(Refer Slide Time: 42:24)

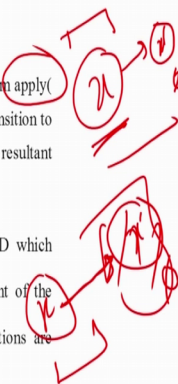
Symbolic Model Checking: EX

- $B_{\vec{X}}$ is the OBDD representation of the transition system that involves variables from \vec{X} and \vec{X}' .

- The OBDD $B_{\vec{X}}$ contains all the transitions of the model. Once we perform $\text{apply}(\bullet, B_{\vec{X}}, B_{X'})$, then it will give us those transitions which are making a transition to the set of states represented by $B_{X'}$, and the variables involved in the resultant OBDD are from \vec{X} and \vec{X}' .

- The entire operation $\text{exists}(\vec{X}', (\text{apply}(\bullet, B_{\vec{X}}, B_{X'})))$ returns an OBDD which involves variables from the vector \vec{X} . It makes the BDD independent of the vector \vec{X}' , and giving us the set of states from where those transitions are originating.

- Therefore, $\text{Pre}(X)$ gives us the states which can make a transition into set of states X .



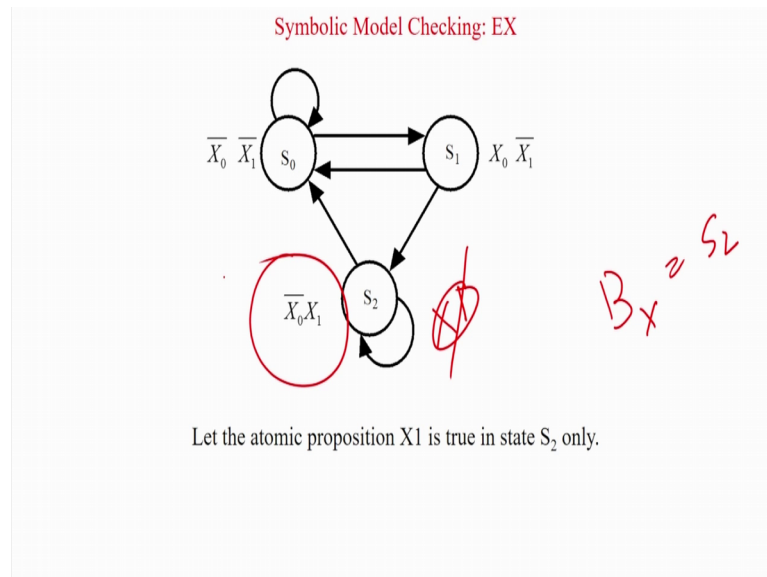
Now, basically what you have to be say when you are going from the apply operation. So, what is it is doing basically apply operation is trying to get your state called as I told you $X \wedge \bar{X}$ and this ϕ is equal to true and it will have variables from both X and \bar{X} because, at this state transition representation is inverting both prime and nonprime versions.

Now, we have to just bring it this state, that is the state from with the next state is X prime with ϕ is equal to true; that means, we have to somehow eliminate the prime variables. So, that is why we are going for the exist operation on the primes, that is we are going to find out transitions which needs to 1 in the BDD which is irrelevant of the prime variables, that is $X \vee \bar{X}$ 1 prime can be any variable, but that needs to be 1.

So, basically if you are going for this 1 it will make the BDD independent of all the prime variables. So, therefore pre of X by this formula will give me the state this 1, slightly involve this thing only you will get the idea that is first we are trying to find out a subset of states called $X \wedge \phi$, where the \bar{X} will ϕ is equal to true then if you are going to apply this state subset BDD true multiply with the transition, subset of the transition represented BDD of course we are going to have such transitions which lead to a state call X prime means ϕ is equal to true ; means what we are going to get after this apply operation, but and this is actually your x .

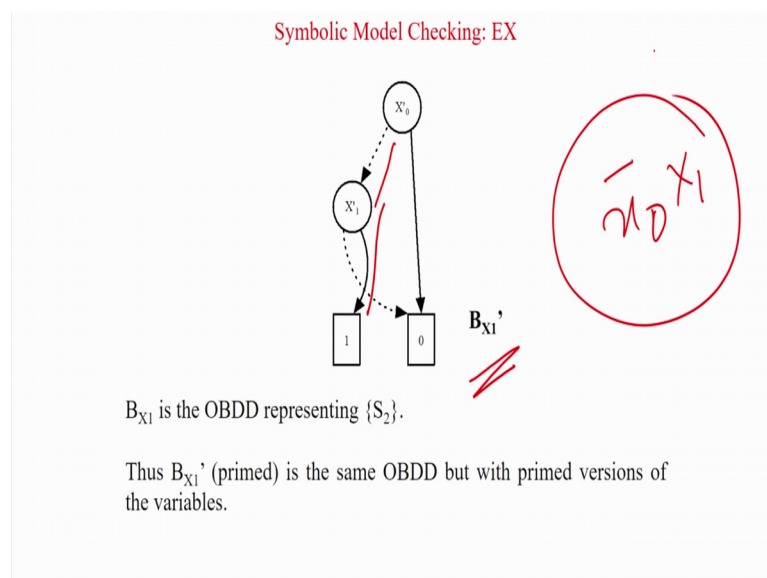
Now, we just want to this and not the other part this one we have to eliminate, so this is done by basically your exists operation we exist operation as I told you we eliminate out some of the variables which you do not want to the in that one, so all the prime variables are you know then you are going to get the answer.

(Refer Slide Time: 44:01)

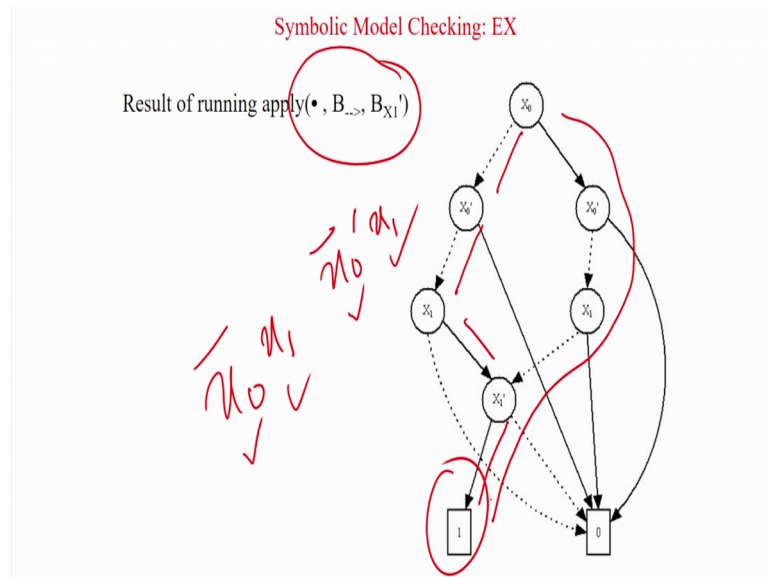


Example this one we have to slightly go back and look at the literature or the text will be uploading on the exists operations, restrict we have already discuss and then it will be very clear to you. So, now let us assume that some proposition called X_1 capital X_1 or for the time being it can also called ϕ whatever you want is true only in state S_2 .

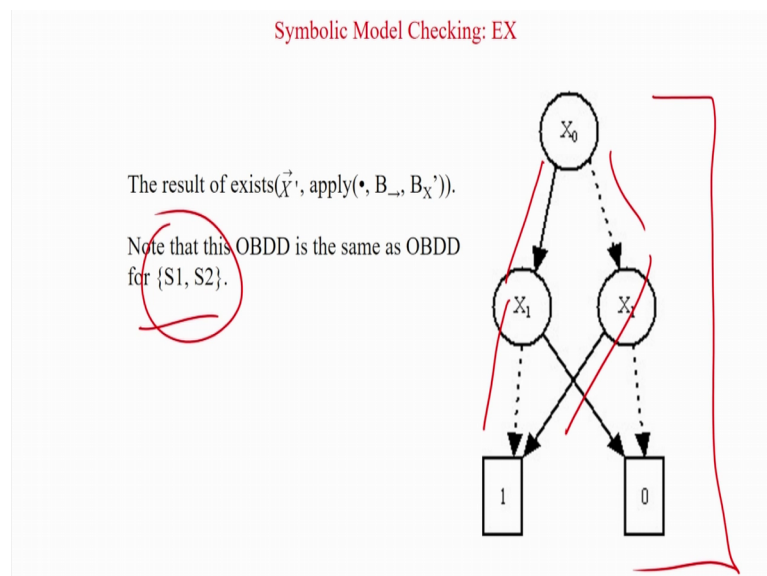
(Refer Slide Time: 44:18)



So, what I have to do I have to represent your BDD x is nothing, but your S_2 . So, this is the subset which is actually B_x , so if you see X this is actually nothing but X_0 prime, see this is x_0 prime and x_1 and x_1 . So, basically 0 and 1 is there into 1, so this subset is represented by this 1 and of course as I told you we have to go for a representation of the next state because it is We require something like this sorry we require finally.



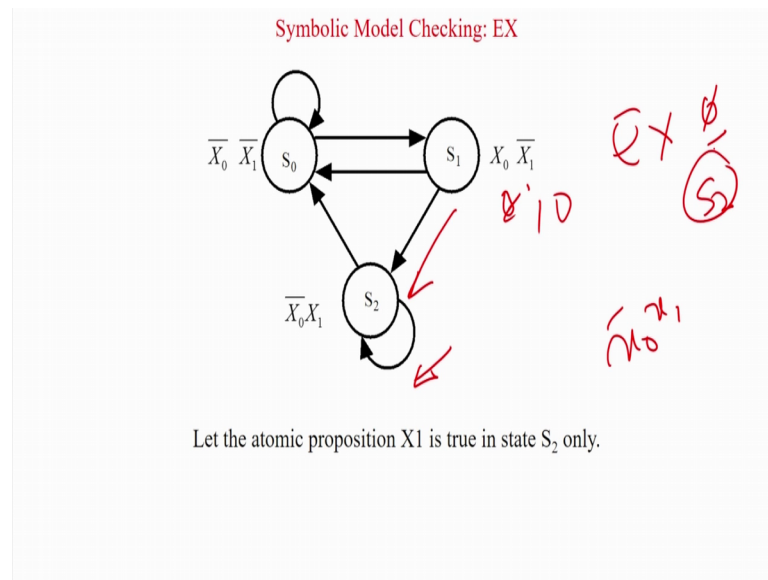
(Refer Slide Time: 45:10)



So, anyway I am not going to again it is in just show below in maybe we can just see there is a self loop between 0 0 to 0 0, this self loop just we see that whether it is here so 0 00 0. So, if you look at it. So, it is 0 0 0 0 so it is going to be 1 that means, this transition is represented by this part.

So, basically this is nothing, but the whole transition representation of these whole Bookie automata, if you can recollect and you can see that will be represent in this manner I have just shown you 1 part which corresponds to this self loop. So, now we have this basically, next is nothing but you have to make a product. So, what is the product this B transition and this state x.

(Refer Slide Time: 46:04)



So, that means this is your state and the whole state transitions your product tool. So, what you are going to have you are only going to have this two transitions into picture. So, let me just see if you can look at it this is nothing but $\bar{X}_0 \bar{X}_1$ and self loop.

So, basically next state is S₂, so only that those things should remain because we have producing S₂ next state with the whole transition diagram. So, it is nothing, but \bar{X}_0 just let me let me correct it $\bar{X}_0 \bar{X}_1$ in the single \bar{X}_1 and of course the next state was present state \bar{X}_0 and \bar{X}_1 .

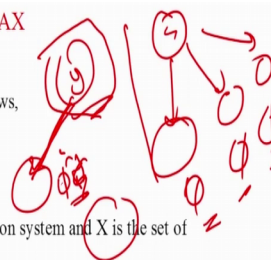
So, that is the self loop that will remain because the next state is S₂ that is what we are trying to look, because our goal is next state is S₂ and here producing with the whole set of transition.

(Refer Slide Time: 46:48)

Symbolic Model Checking: AX

How to Compute $\text{Pre}_v(X)$:

$\text{Pre}_v(X)$ can be represented in terms of $\text{Pre}_3(X)$ as follows,

$$\text{Pre}_v(X) = S - \text{Pre}_3(S - X)$$


- Consider S is the set of all the states of the transition system and X is the set of states of our interest.
- $(S - X)$ gives us the set of states that are not of our interest.
- $\text{Pre}_3(S - X)$ returns us the states that are having at least one transitions to the set $(S - X)$, and we are not interested for those states.
- The states which do not have any transition to $(S - X)$ are of our interest, and so the remaining states of S are the correct choice for $\text{Pre}_v(X)$.

pre 3

So, only two transitions will remain like in this case this is your next state which is your sets subset and then you are producting with the n this is your next state which has to be true and you are producting with all transitions only this and this two transitions qualify. So, let me just trace 1 transition so $X \ 0 \ 0$ prime then $X \ 0 \ X \ 0$ prime and $X \ 0$ bar is this way then $X \ 1$. So, we are going to here and then basically $X \ 1$ bar equal to 1 so this is the 1 and you can see this is the 1, so that self loop is represented by this 1.

Similarly, you can track out all paths which is reading to one by this methodology right, this is one path will be there and this will be another path which will be corresponding to this only two path should be one, because only two transitions qualify. So, this is one path which I have already shown you and there is another path. So, there will be only two paths reading to one which is capture by the BDD

Now, what we have to do so this basically represents the two transitions where the next state is S_2 , where ϕ is equal true next we have do is very simple you have to eliminate out all the prime.

(Refer Slide Time: 47:55)

apply(\bullet , B_{\rightarrow} , B_{X1})

The diagram shows a directed graph with states x_0, x_1, x_2, x_3, x_4 and actions 0 and 1. Red lines indicate a policy that chooses action 1 from x_0 and action 0 from $x_1, x_2,$ and x_3 .

So, this is nothing but the state S1 and S2 you can easily see. So, this 1 will be nothing, but your state S1 and S2, if you just do the apply operation over exists operation over then you will find out, so S1 and S2 will be there. So, like if you want to see this is 1 0. So, let see 1 0 we can track so there will be two states basically, so 1 0 this is 1 state and this is 0 1 so S1 S1 and S2 are the two states, so two paths which is 1 and all as a 0.

So, these by this where we have shown how EX can be easily operated or easily obtain from model checking from symbolic model checking, without explicitly modeling the bookie automata in terms of states we modeling's in terms of BDD and we go.

Just before we will close how to go for the for all, this very simple just the dwell of it. So, what we do is that so basically we take S the entire set and then what we do we take the pre of S minus X and we do it. So, what is the idea basically S is all the set of states

of all transitions X and X is the subset of states under interest; that means, we want to find out say these are the states and the where ϕ is equal to true and we have to find out of state with all transitions are going to this one. So, rather what we do we try to find out some states y , where one transition is basically going to some other states where ϕ is equal to false that is the ϕ prime.

So, we are going to find out that is nothing, but your S minus X , we are trying to find out all states basically from where this is a transition which goes to some other state, where ϕ is not equal to true then we go for the same existential operation, that you are going to trying to find out all states from where there is a at least 1 path where ϕ is not equal to true; that means, those states should not belong to this category because, the category all states from all states from where all paths reading to next states where ϕ is equal to true.

So, what we will try to do we will try to find out all states because we know how to solve the existential quantifier because, we know that preexist X we already know how to we will be using this to find out this is for all. So, what we are trying to do we will try to find out all states y where there is at least one path to a next state where ϕ is equal to false and that is the states which does not qualify our requirement S minus X will give you the answer.

So, that is what is the written in the slide very simple just the dwell of it you can easily find this out, because already we know that existential is simple to solve and how we can find out for all, that is states $S \setminus X$ where for where in all for all the next state ϕ is equal to true all transitions.

So, just we do in the negated way we will find out all states from where there is at least 1 path because existential we know where ϕ is not equal to true, though those you remove from S job is basically done. So, that what is what is written in the algorithmic states. So, basically this brings us to the half part of the lecture unsymbolic model checking because, already we have seen $\exists X$ and maybe or $\forall x$.

So, this one actually corresponds to $\forall x \exists X$ means there exists a path when in next state ϕ is equal to true, the other one this one actually corresponds to $\forall x$ all paths when next state ϕ equal to true. But there are so many EU au future global, so many other constructs where there LTL means temporal formulas are there we will next class we will

try to see all of them and how we can solve by the same approach, modeling the states substrates where ϕ is true then producting with your basically your transition function and then exactly what the algorithm has to be followed.

Thank you.