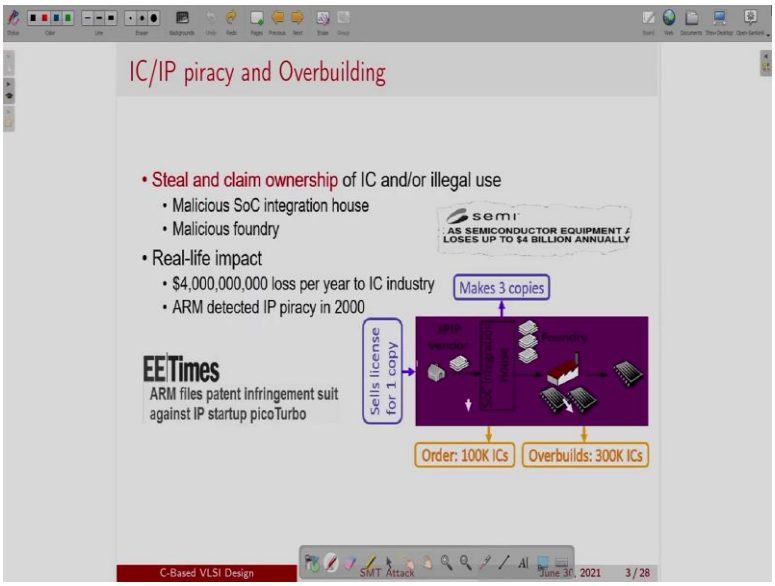
C-Based VLSI Design
Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 11
Securing Design with High level Synthesis
Lecture - 38
Attacks on RTL Logic Locking

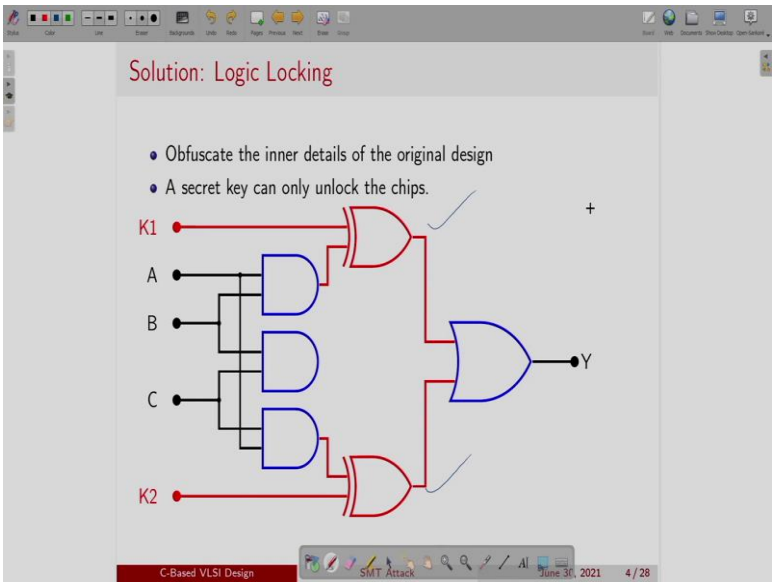
Welcome everyone. In today's class, we are going to discuss on an SMT-based Attack on Register Transfer Level Locking. (Refer Slide Time: 00:59)



So, in previous classes, I have discussed that this IC industry facing a huge security problem because of this third party, outsourcing of their integrated property to third party for fabrication and it is because this fabrication cost is huge. So, most of the time, you develop your IC; you invest all of your effort, you test, verify, make sure it is ready and then, you give that layout to third party to produce the IC for you.

But now, the question is that if you give it, you are exposing your IP to somebody else and he can actually pirate your IP and he can create a duplicate copy, it can use it for some other purposes or it can overbuild. So, and as a result, you will lose your revenue.

(Refer Slide Time: 01:46)

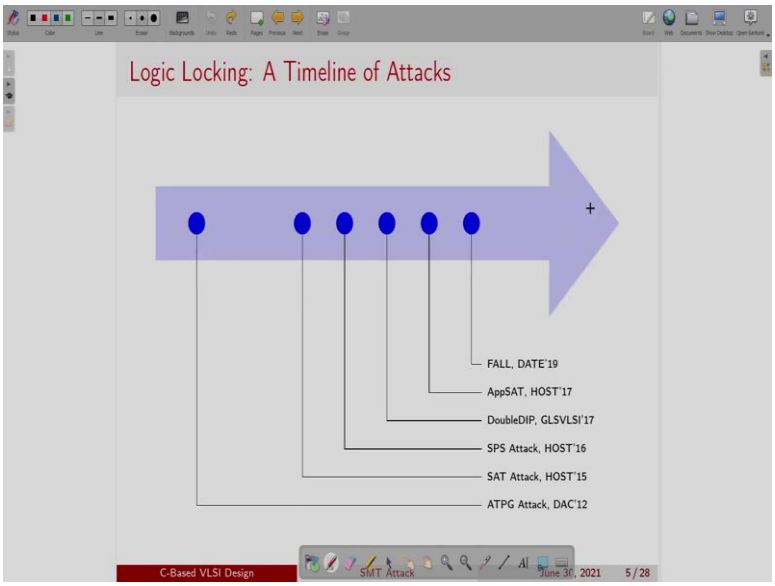


And we have also discussed that to stop that the most important technique is logic locking and concept is very simple is basically simple; but very powerful is that you have your circuit, you add certain extra circuitry, you add some key just like here. So, you have the original circuit earlier and then, you add certain key you add this xor gate so that this is now locked.

So, now what is going to happen here is that you have this, you are giving this layout to third party; but you are not going to give this key to them. As a result, even if they have this circuit unless they know the correct value of this K1 and K2, you they won't able to use the IP for any purposes.

So, and now, if you think about a circuit where you give 200 or 400 number of keys, it is impossible for them to identify what is the correct key combination and for only one possible value of this key, this circuit will work and for rest of the 2 to the power n minus 1 possible value, it won't work correctly. So, that is the very basic idea of logic locking.

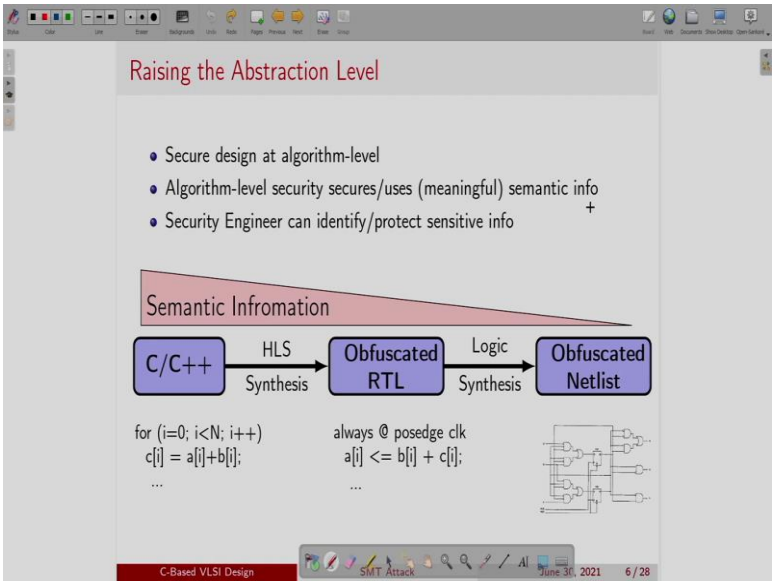
(Refer Slide Time: 02:51)



So, and then, we have seen when I have already discussed that there are blither of work is going for last 5 years or so, where people are coming up with various kind of laid technique and then, there are another set of work researcher, they try to break those attack. They are saying that even if you lock this way, this is the technique if I use, it will break the circuit. So, these are lot of works is happening in these directions.

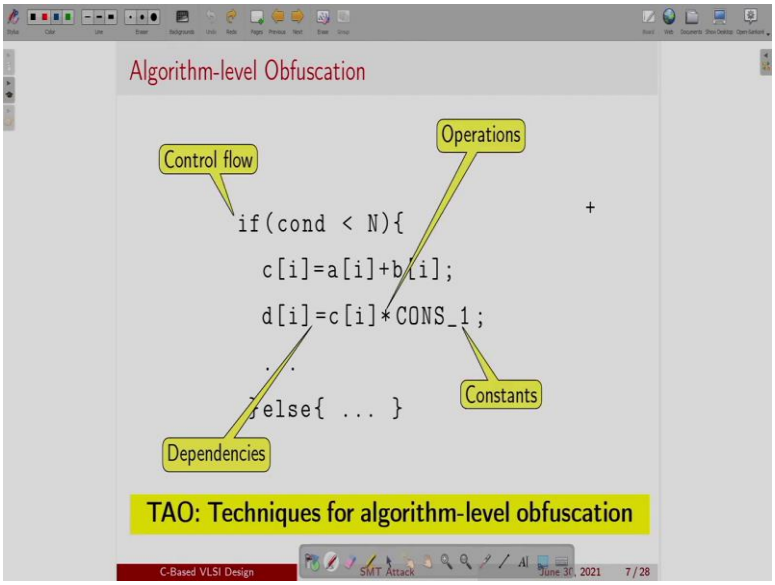
And most of this works actually target the gate level circuit. So, all these logic locking techniques comes in the gate level designs and there are two kind of technique; how do you lock the sequential element, how do you lock the combinational element. And even similarly, for attack part also, people try to attack this combinational part and sequential part kind of separately.

(Refer Slide Time: 03:36)



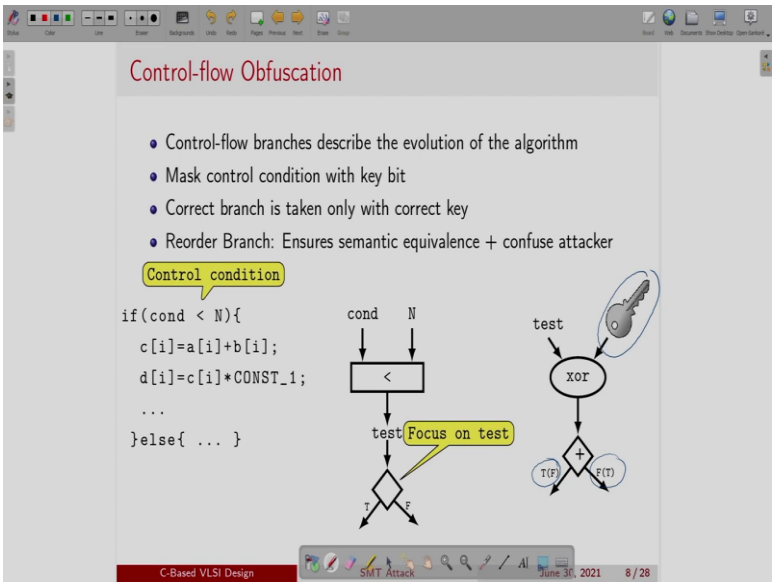
Then we have discussed that in a recent work is called TAO, they try to lock the circuit during high level synthesis. So, advantage of this TAO is basically you actually lock your circuit early in your design flow and because at the higher level of abstraction, you have control information in you have in the design, your locking can be more powerful than the structural locking what you do in the gate level. So, that something is really interesting.

(Refer Slide Time: 04:06)



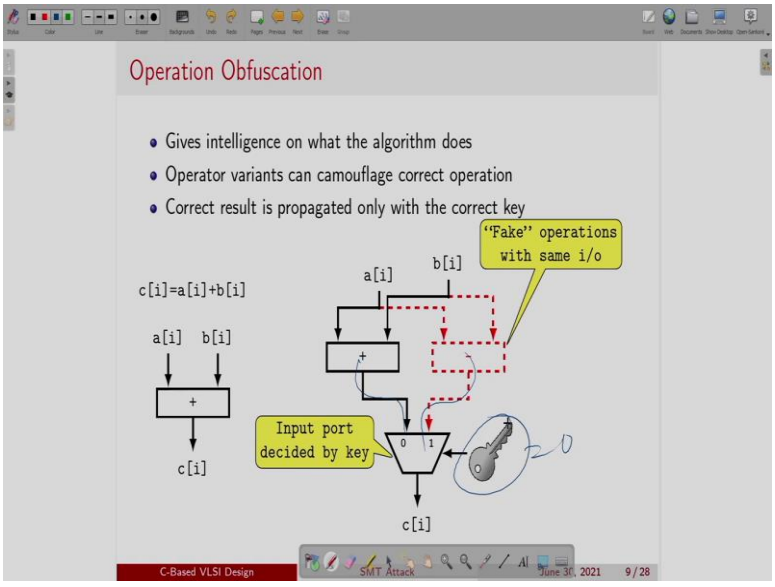
And we have already discussed that this tool called TAO, which call an algorithmic level obfuscation usually target three things; it try to lock the control flow, it try to lock the operations, it try to lock the constant, it try to lock the dependencies that we have already discussed. I will just go through quickly.

(Refer Slide Time: 04:23)



So, basically for this control what you they have done is basically whatever the actual condition checking is happening, it will just add one more level of xor with the key. So, now, based on this key value if and else condition which there. So, you may have to swap this if and else block if based on the key value. But since user does not have the original behavior with them. So, they do not know what is the behavior; what is the correct, if else condition here.

(Refer Slide Time: 04:55)

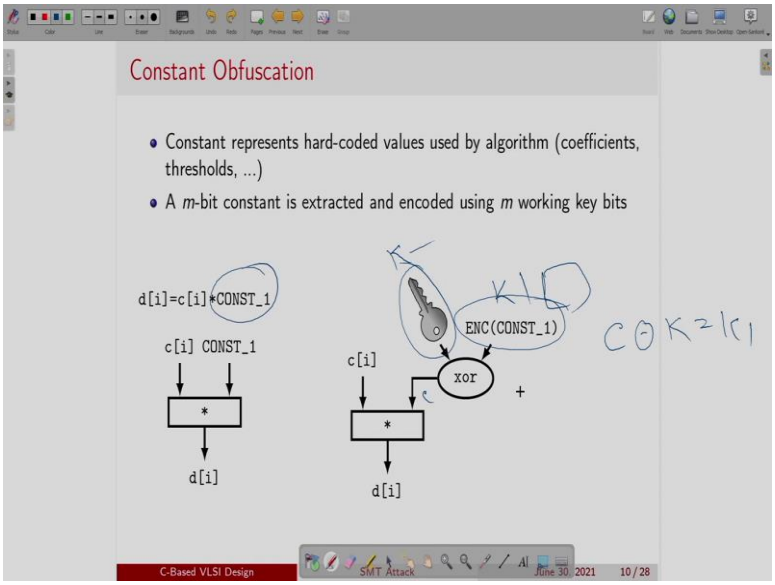


here.

Similarly, this operation of obfuscation is very simple that you just use a key and it will select the actual operations or some junk operation and here, I have seen the key value is

0. So, if the key value is 1, then the actual operation will come in the mux the other side. This is what is operation of obfuscation.

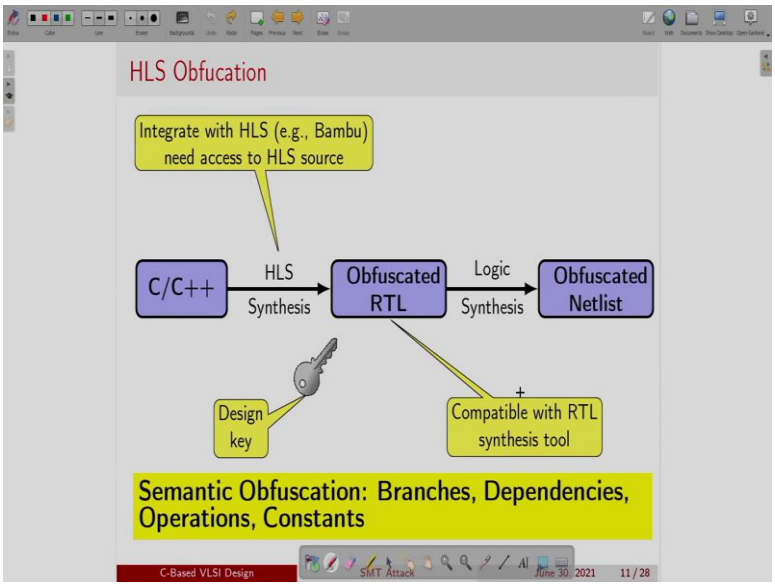
(Refer Slide Time: 05:15)



And then, also we have discussed the constant obfuscation basically you just you take the constant of your behavior and you just xor it with some key. Say if the 32-bit constant, you use a 32-bit key and then, you just xor them, there the encoded things is stored in the hardware and then, so this is basically the encoded value.

So, if the value of constant was c, your key is K. So, and this is K1. So, you are going to store K1 here and this is your key. So, your key will come from it will then this is basically your c. So, this is how the whole function will work. Since you do not know this key value, you won't able to guess what is the correct value of c.

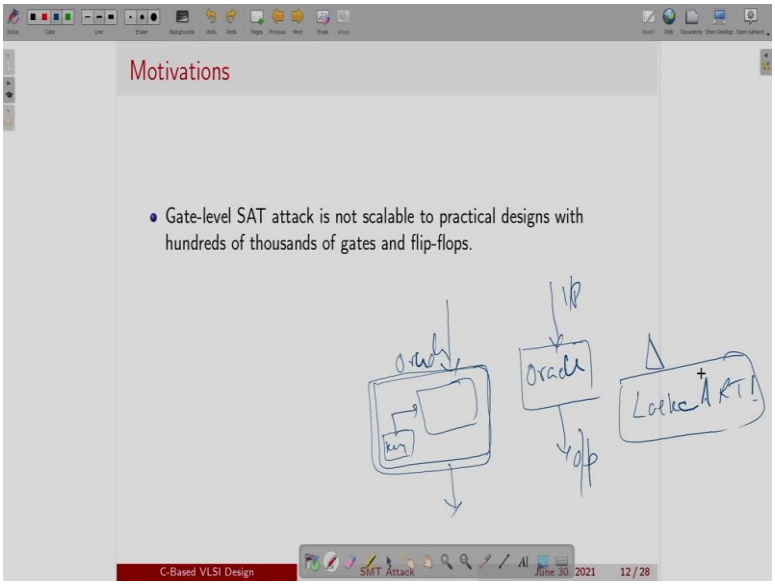
(Refer Slide Time: 06:01)



And similarly, the dependencies are also you can do that. So, this is what this has been done during high level synthesis and since it is has been done on the high-level synthesis that is of our interest. So, basically, the things that I am discussing in this week is how can we make a secure design through high level synthesis. So, that part I have already discuss and TAO is one of the powerful techniques for that.

And today's class, what I am going to see, is this particular locking is kind of robust; is it possible to break this? Say for example, that means, I given an RTL, where this circuit is locked with the keys. Can I identify the correct value of key using some method? The answer is yes. So, there is a recent work that we propose, we show that we can actually break this particular locking.

(Refer Slide Time: 06:53)



And what is the attack model there? So, let us let me try to explain, what is the attack model. The attack model is something like this. So, you basically have this oracle. So, this is the oracle. What is that?

So, you buy an IC from the market and if you give a input, it will give you the correct output; only thing you cannot see inside because this is a IC and how this oracle is actually built so that this is work how this oracle gets the correct key? It is basically if your circuit is like this, this is the circuit and there is a non-volatile memory, where you actually put the key inside the circuit.

This is your oracle and this key comes from here and since this is we cannot break it; you cannot get into the key. This is what I buy from the market. So, this is my functional oracle. So, I can always buy it from a market. Since the correct key values already stored inside this oracle in this functional IC, I will get the correct output.

So, input output correctness things, I can actually always get it from a functional oracle and also, I get a locked circuit which is basically a layout level thing and there are so many methods has been proposed to reverse engineer the things.

In in our one of the classes, I also talked about how can you reverse engineer a C code from RTL. So, that we have already discussed. Similarly, from the layout, you extract a gate level design from the gate level design, you try to lay extract an RTL design and since somehow somebody has done that hard work and you it is given you that RTL.

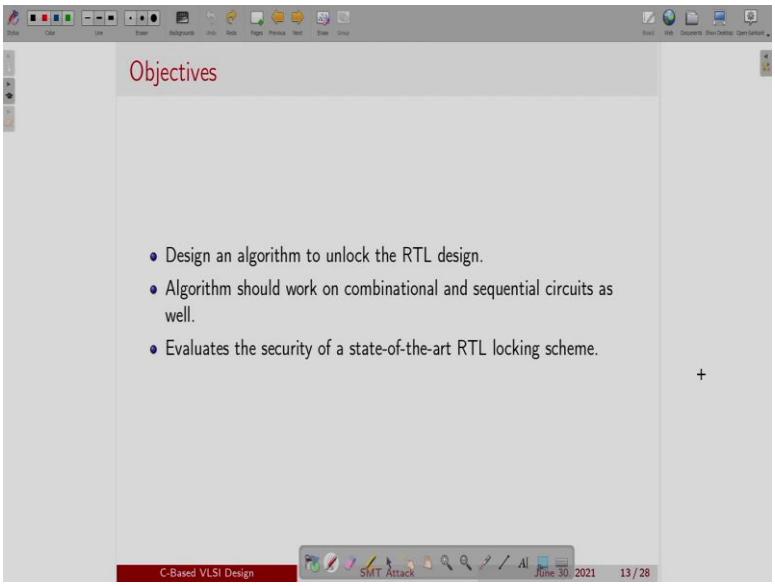
So, now, you can see the RTL that is a text file with you, where you can see how many keys are there, where they are actually used; so many things you can see that. Only things you do not know the value of those keys. So, these are the two things you have with you. So, you have a locked RTL. So, you have this locked RTL. So, that is with you and you have this oracle.

So, these are the things is with you know you want to apply certain technique, so to identify the value of the key value that is used in the locked RTL that is the problem. So, now, one might argue that why we have to develop a method at the RTL level? Because there are many techniques already getting proposed at the gate level circuits.

So, the point here is that in the gate level circuit, if you just convert an RTL circuit into gate level circuit, it will be at least ten time more gates and all. So, it is a 10 time more complex. So, analyzing the things at the gate level will be bit difficult because it is a bigger circuit and another big problem is in the gate level circuit as I already mentioned, we usually analyze the combinational circuit and sequential circuit separately.

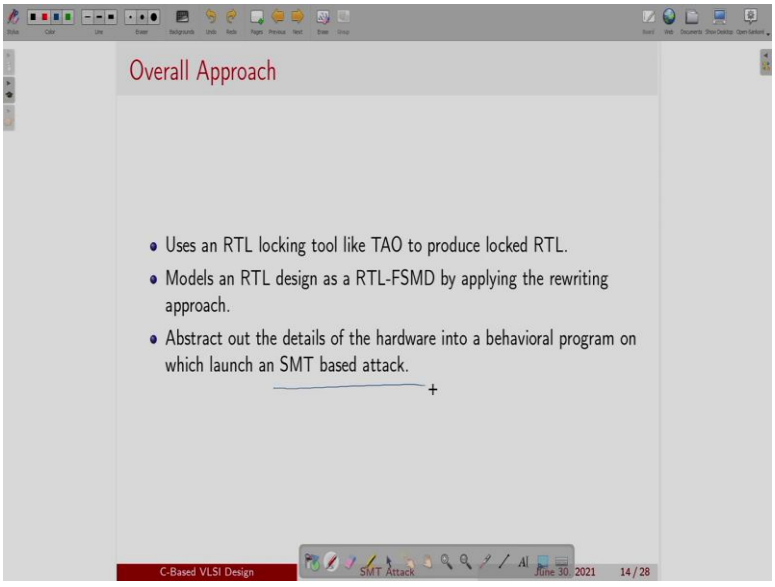
But here the locking has happened at the control flow level. So, we do not know how they are going to impact both sequential and combinational part, if I synthesize the circuit into RTL. So, it may be possible that the gate level circuit, the gate level attack that is already proposed may not be used directly. So, and it will create lot of problem. So, that is why we try to propose an attack technique at the RTL level itself that is what I am going to discuss in today's class.

(Refer Slide Time: 10:13)



So, what is the objective here? I already have this as I mentioned, I have a function oracle that I buy from the market running chip and I actually reverse engineer an RTL locked circuit with me and then, I try to see can I break this circuit or can I identify the values of this key. So, that is our objective.

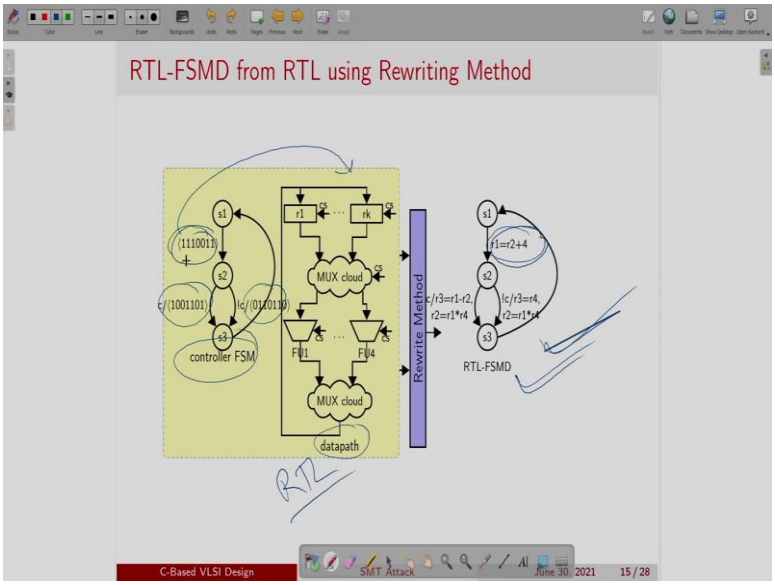
(Refer Slide Time: 10:35)



So, what I am going to do here is very interesting that the RTL to C reverse engineering technique that I have already discussed in some of the previous classes, I am going to utilize that particular thing here. So, the advantage of applying that is that I have the RTL which is generated by some high-level synthesis tool or the which is basically extracted

from that layout, I have the RTL. And since this RTL was generated by high level synthesis tool, it has a separable controller path and data path.

(Refer Slide Time: 11:18)



So, I can apply that RTL to C reverse engineering technique and I can actually extract a high-level behavior. Just to recap what we have done actually is you have that RTL; this is the RTL that is generated by high level synthesis tool which has a separable controller and data path that I have already discussed.

And the idea was that in every state, the controller gives some 1 0 signal to the data path and so that certain operation is getting executed. And then, our idea is that I try to analyze what is what the operation is happening because of this control signal assignments and if I just analyze and identify, then I can replace this control signal by the corresponding operation that is happening.

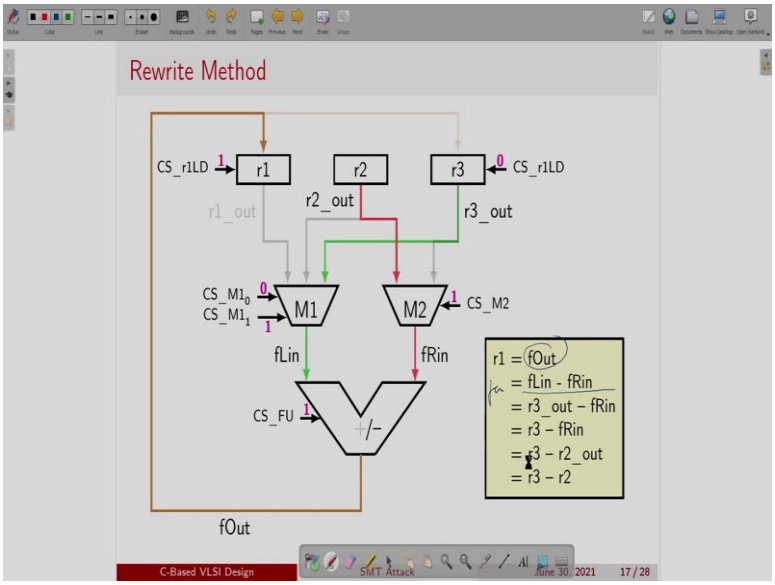
And if I do it in every state, I will end up getting this behavior and this behavior does not have any clock, does not have any multiplexer, de-multiplexers, any detail of the RTL. This is an abstract behavior. This is an RTL finite state machining data path. This is a high-level behavior; I can represent that as C that that is that is what I have talked in earlier classes.

But I am going to use this model. So, this is a high-level behavior and why we reason about this? Because as I mention this locking that happened during RTL, this high-level synthesis specifically the TAO, it actually tries to lock these operations, conditions and this constant.

So, if I get this kind of high-level behavior, it will be easier for me to create an attack on that. So, this is the first step that I am going to do is I am going to extract a high-level behavior from the RTL and just to recap how these operations that is going to happen in

the data path because of a control assignment is done by a rewriting operation, just I have an example here.

(Refer Slide Time: 12:58)

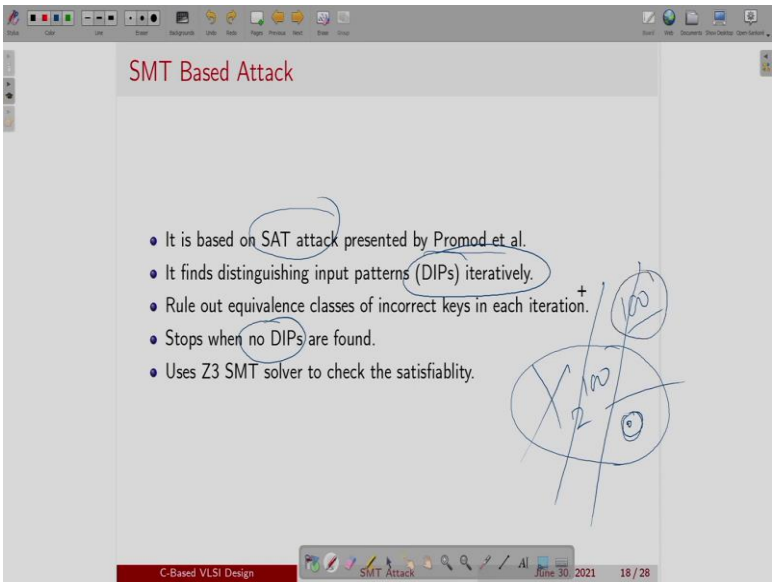


So, suppose this is the data path and these are the control signals are there in the data path that is the value will come from the controller and say in one clock, you actually give these values to the controller some 1 0 values. As a result, only few of the operations will be active.

So, these are the active micro-operations and then I am going to identify the operations, where the left-hand side is a register. So, I can understand this is the operation, where the left hand side is register; from there, I am going to do a rewriting process, I will just start keep rewriting these operations with another operations, where left hand side was fout.

So, this is how I am going to do and finally, I will get an operation which is r1 equal to r3 minus r2. So, this is what I have already discussed in detail, so I am not going into detail of that. So, the idea of our technique is that we will extract this FSMD and then, I am going to launch a SMT base attack on this FSMD. So, this is our overall approach. So, we will go in to detail now how this SMT base attacks works and our objective is to break TAO, whatever the technique that TAO imposes that a constant obfuscation, operation of obfuscation and then, this condition obfuscation.

(Refer Slide Time: 14:33)



So, I am going to break these three kinds of techniques using a technique called SMT based attack. So, from the RTL, I abstract out of an FSMD and then, I am going to launch the attack that is what the overall process.

So, now, what is the SMT based attack? So far, I have the RTL-FSMD that is already extracted from the RTL and it has its basically locked the key values are already in that behavior. So, this particular attack is basically inherited from the SAT attack that is proposed by Pramod Subramanyan et al. So, that SAT attack is actually proposed for the gate level circuit.

And since, we have a high-level behavior, where we have integer, floats and all know be kind of data path, so instead of SAT attack, we are going to use the SMT attack. So, SMT is the same thing. So, both SAT tool and SMT tool is basically is a theorem prover basically, where you give a formula. This tool will say whether that formula is satisfiable or unsatisfiable that is the only things that is tool does.

And SAT tool basically it takes in the Boolean variables, SMT you can give integer floats and all that is the difference. So, you have a formula which is if it is a Boolean formula, you just call a SAT solver and it will say this formula is a satisfiable or unsatisfiable.

And similarly, SMT solver, you can give a formula which is where the variables can be Boolean, integer or real and you can ask the solvers to see whether that formula is satisfiable or not. So, that is the tool we have and we want to use that particular tool to identify the key because this tool does not understand what is locking, what is what; but only thing is that the way I have to model the problem such that this formula becomes satisfiable, it has some meaning in my context and if the formula become unsatisfiable

that has some interpretation in my context. So, I have to model my attack problems in a formula, where the satisfiable and unsatisfiable has a distinct meaning in my context.

And so, the core idea of this SAT attack or SMT attack is finding the DIP iteratively. So, this is what very important powerful concept here. It is called distinguishing input patterns. So, the basic idea of the whole SAT attack or SMT attack is that every iteration, you find a DIP.

I will come to that what is DIP and then, based on the DIP, you if you find a DIP, you basically add one more constraint in your formula. You basically have a formula; you add one more constant to the formula so that the search space of that DIP will reduce gradually.

And then, every iteration, you keep doing this and once you do not find a DIP; that means, you are done and so, then the key remaining in the search space of the key is the correct key. So, that is the overall approach.

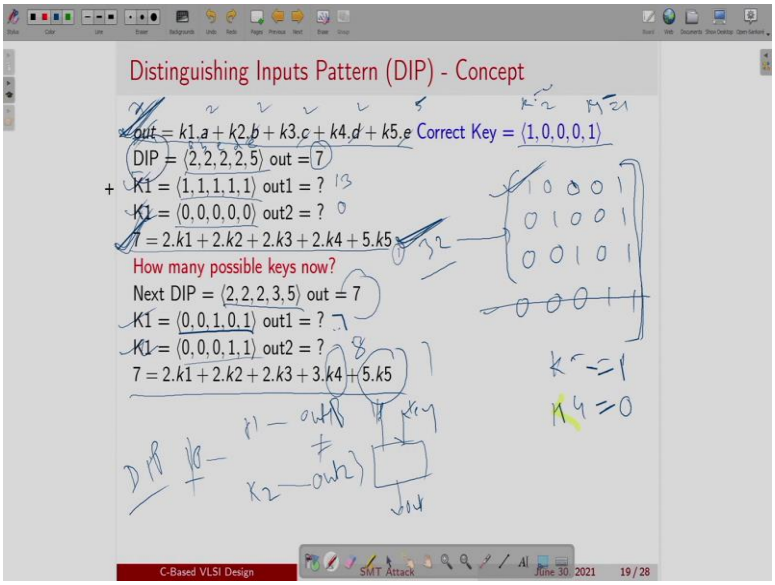
Then, then what we have to understand that what is DIP and how does it help to identify the correct key. Because I am going to create a formula and give that formula to the SMT solver and it will just give me a DIP and so, it will say that formula is satisfiable and this is the input for which this formula is satisfiable and that is my DIP.

And after sometime this tool will say no there is no DIP exist and then, I will show then this is the key that is the overall idea of identifying the keys using SMT attack. So, first thing we have to understand what is DIP and how does it help in reducing the key search space.

So, you have to understand that, if say there are 100 keys. So, there are 2 to the power 100 possible values; see huge search space and only one of the values is correct key. So, I have to basically search this whole search key search space and I have to identify one key and this DIP is basically help you to reduce these search space very fast.

So, maybe one DIP just reduces half of the keys, just remove half of the keys from the search space. In the next DIP probably removes another part of this. So, then this DIP will remove this. So, within say few iterations, I will identify the key. So, I do not have to go for 2 to the power 100 iterations rather I maybe say 10, 12, 15 iterations, I can identify the key for even a search space like 2 to the power 100 or 2 to the power 200 that is the idea of DIP. So, let me try to explain what is DIP.

(Refer Slide Time: 19:05)



So, what is DIP, it is distinguishing input pattern, is very simple thing that I have a locked circuit. I will give an input and key values can be different. The key can have various possible values.

There should be at least two possible values of the keys, say one possible value is k1, another possible value is k2; I will get output 1 from the circuit for key k1, if I give the value k2, I will get output 2 and it says that these outputs are different.

So, this input is my DIP, if I take this input value; remember this locked RTL has two inputs. The actual input and the key and the output. So, in the locked circuit, you have to give some random value of the key. So, the DIP says that this is the input for which if I consider two values of the key and for that, I will get two different outputs; the output will be different. This is as simple as that.

So, what does it mean? Since I am going to get two different output, so at least one of the output is wrong because actual circuit has one correct output; both of them may be wrong or one of them may be wrong, but both of them cannot be correct because the circuit can have only one correct output for a given input.

So, that means, it guaranteed that since I get two different values of the output, at least one of the keys is wrong and that will be excluded. If both of them is wrong, both of them will be excluded; but in in reality, it is not 1 or 2, it is basically class of keys will be removed because of this DIP.

So, let me just explain here. So, I just take a simple example, hypothetical example. Suppose this is my locked circuit say or the locked behavior which is basically out equal to k1 into a plus k2 into b plus k3 into c plus k4 into d plus k5 into e , where this a, b, c, d, e is the inputs and k1, k2, k3, k4, k5 are the keys.

And let us say the correct key is this 1,0, 0, 0, 1. See this is my k1, this is k2 and this is k5; k1, k2, k3, k5. So, this we do not know, I want to identify that key; but for explanation purpose, I mentioned the correct key here. So, if I give this, so I have this locked circuit and I have a functional oracle which will tell us that if I give some value of a, b, c, d, e, what is the correct output? So, I say that one of the DIP is 2, 2, 2, 2, 5 , where this is a, this is b, this is c, this is d and this is e. So, I just give the value of a equal to be 2, b equal to 2, c equal to 2, d equal to 2, e equal to 5.

Since this correct key, I know that k1 equal to 1 and k5 equal to 1 and rest are 0. So, basically that means, a plus e; the correct output will be a plus e. So, which is 7. So, if I just give this to a functional oracle, this 2, 2, 2, 2, 5, it will say the correct output is 7; although, I do not know how it how it we obtain that because that inside I do not have because that is oracle.

So, now, why this is DIP? Because if you just consider this k1 is basically all 1 and this is k2 which is basically all 0. So, if I just put all 1, what will be this value? So, it will be 2 plus 2 plus 2 plus 2 plus 2 plus 5. So, it will be 13. If all 0, output will be 0.

So, definitely this this 2, 2, 2, 2, 5, this is the input is the DIP because if I two consider two different values of the key, I get two different output 13 and 0. So, that means, it I will to distinguish the keys. So, that is why this is distinguishing input pattern. So, I understand the concept of DIP definitely. So, that is this 2, 2, 2, 2, 5 is a DIP for this example.

Because if I two consider two random values of k1, k2; I get two different outputs. So, here since there are five keys, so the possible value is 2 to the power 5, 32 possible values are there in key and I just consider 2. So, initially, that tool has to identify two such key and it say that for this input, for these two key outputs is different.

So, remember here is there is no unique DIP, there are many DIPs. So, you just consider any value of this a, b, c, d, you can consider all any two a key, you will get a different output at least for this example. But the tool that SMT solver will give me that I will talk about that how I will ask this as a formula to the solver; but let us assume that somebody tell me that this is the DIP.

So, now how this DIP helps that is the things is important here. So, what I do that with this key, I know what is the correct output. So, initially, this is the formula I have given to the tool. Now, I am going to give one more formula. What is the formula? Because I know that the value of a, b, c, d and e is 2, 2, 2, 2, 5.

So, I am going to replace this a with 2, b with 2, c with 2, d with 2 and e with 5 and I know the correct output is basically 7. So, I will get an equation and that equation is this that 7 equals to 2 into k1 plus 2 into k2 plus 2 into k3 plus 2 into k4 plus 2 into plus 5 into k5. So, that something an additional constant.

So, initially, this k1 to k5 can have any value; any, one of the 32 values. But I am saying now the k1, k2 satisfy this equation as well as the k1 to k5 will satisfy this equation as

well. So, now what does this mean? So, it saying that the. So, from this if you understand, since how many ways, you can actually get this 7 because these are all even number; this 5, this must be 1 and only one of this value will be 1.

So, here you have to understand this k1 to k5 are Boolean variable. So, you can understand here that now earlier, there are 32 possible keys are there; but now, I have added this particular equation and it actually ensures that only 4 possible values are now possible is basically because you have to this k1, k2, k3, k4 and k5 value such that their summation will be 7.

And I argue that here the possible value is basically 1. So, one of the values will be 1 only. So, only one of the values will be 1. So, so the possible value should be 1, 0, 0, 0, 1. This is the correct key or 0, 1, 0, 0, 1 or 0, 0, 1, 0, 1; 0, 0, 0, 1, 1.

So, now you can understand that earlier, there are 32 possible keys. Now, just 1 DIP removed 28 keys from that search space. Now, I have only 4 keys and you can understand that the search space, the correct key also present. So, this is what DIP.

So, DIP makes sure that in every iteration, I am going to get a new constraint which will reduce the search space of the queue by a huge margin and the example that I have given here, initially since there are 5 keys. There are 2 to the power 5, there is 32 possible key values are there.

But after getting this DIP, once I add this constraint, my search space only has only 4 possible values because here to satisfy this k5 must be 1. So, actually I identify the value of k5 basically and one of the value from k1 and to k4 will be 1 because the summation of that will be 7 . So, this I do now. So, I have only 4 possible correct value, then I am going to again ask the SMT solver or somebody, I will say you identify a DIP that will satisfy both this equation now, not only the first equation. So, you have to consider the key value from this set only, not from the other sets.

So, if I ask that, so we will have another DIP is this and again, the output will be 7 because I know the correct key is this and now 2 DIP is basically you have to take it from this set, you cannot take from this set and say this k1 and this is my k2. So, these are the possible value.

So, for this value, it will be output will be 7 and for this value output will be 8. So, this is what the value. So, then this is DIP because I consider two values; k1 equal to this and k2 equal to this and I end up having two different values.

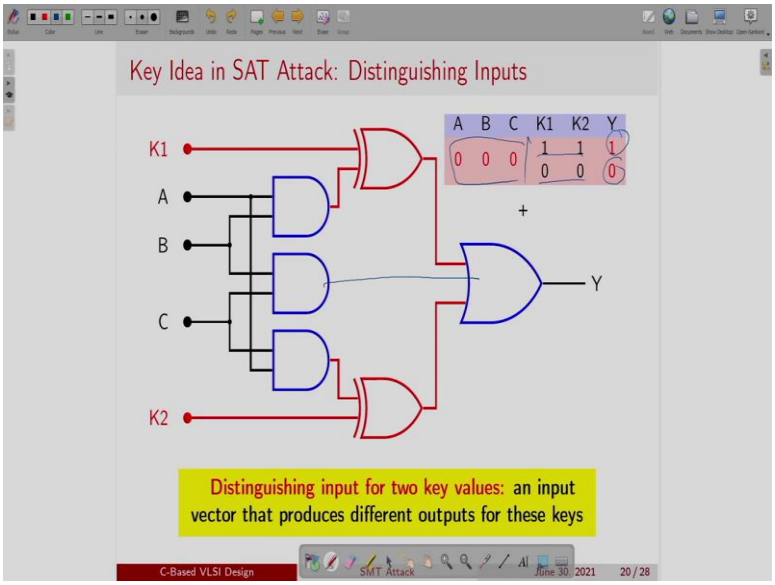
So, this is a DIP and now, I can again, I will just put this value into this equation and I will get this key this equation. Now, we can see here this I already know is 1. So, to get this 7; this cannot be 1 because this is 3, then 5 plus 3 will be 8. So, I already identify k5 equal to 1 and then, k4 equal to equal to 0. So, this is what the advantage. So, now, I got k4 will be 0.

So, now from this, I can actually remove this, wherever the k4 is 1 I remove this key now. So, my search space will have now 3 keys. I will do the same process again and again and every iteration, I am going to remove the keys and probably in the next iteration, I should be able to get the correct key. This is the overall let us say concept of this attack.

So, what are the things we have to do now? We have to model this program somehow, I have to model this program and then, somebody has to give me the DIP. So, these are the things that you have to do.

Once the DIP is there, so somebody has to identify the new constraint and then this constraint will be added to the earlier formula and this whole process will go on. So, here identifying the DIP, we have just used the SMT solver. So, that is the overall idea.

(Refer Slide Time: 30:44)



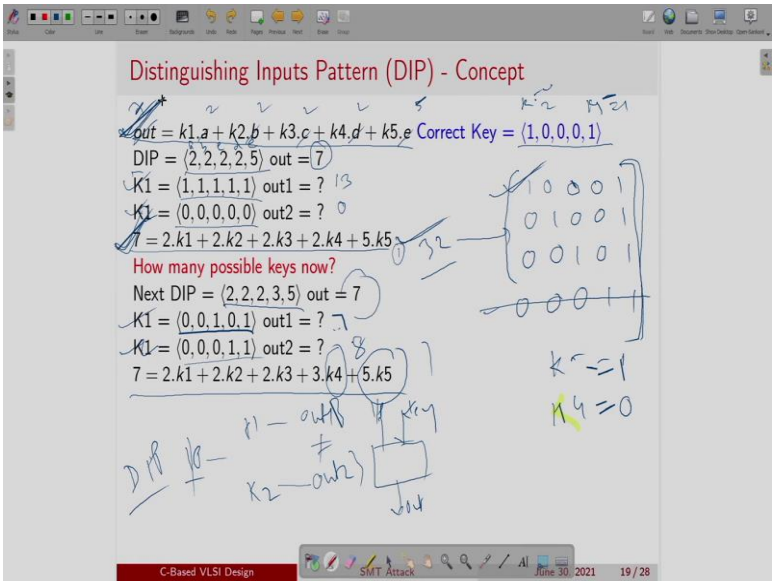
So, quickly, I will go through the things. So, again, the I will take the same example. So, in the circuit level, if I just try to apply what is DIP. So, this is my circuit and this is my locked circuit and if I give this value.

So, because if you can understand here that BC, if you give BC equal to 1, this will be 1 and this OR gate, the output will always 1. So, it does not matter what is the key value. So, for all possible key value, your output will be 1. So, this these are not the DIP. So, this is not a DIP; this is not a DIP. But if I just apply 0 0 0, then it will depend on the key.

So, if I give 1 1, I will my output will 1; if I give 0 0, then output will be 0. So, then this is the DIP. So, basically, the point here is that I try to mention that not all input is DIP. So, I have already shown in the previous slide that whenever BC equal to 1, so that

means, there are two value a can be 0 or 1. So, those are not the DIP; but 0 0 is a DIP. So, DIP is not that all possible input is DIP. So, that is the form the SMT solver will find.

(Refer Slide Time: 31:56)



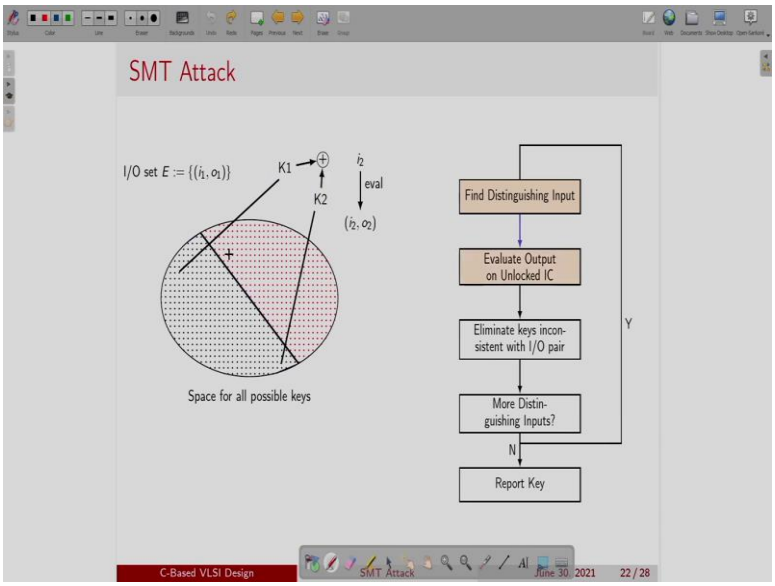
So, and how do you model that is the second question that I ask that how do you model the program as a formula because the SMT solver does not understand anything rather than a formula. So, what we do is basically we have a program say I have a way to model the program.

I will talk about that how and then what I am going to do in that first program, I will replace all the key value with k1 because I want to find out two values of key. And the second copy of the program, I am going to replace the key with k2. So, I rename the variable basically earlier it was said k; I just rename that variable with k1 and k2.

So, and X is my input output will be Y1 and Y2. So, so this is a formula that say model this program, I will come back to that point how do we model that program or the FSMD. So, basically, it says that if I give input X and key input K1, output will be Y1 and this says that if in this circuit or say program, I give the input X. So, X will be common because this is my DIP and if I give the input K2, my output will be Y2 and because this is DIP, Y1 must be not equal to Y2.

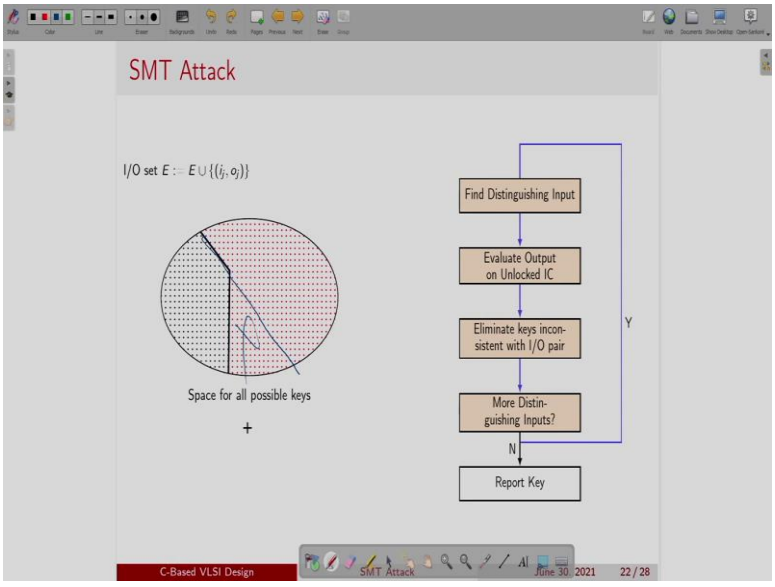
So, basically find X. So, this is the formula, I am going to give it to the SMT solver. So, if I give this formula to the SMT solver, it will identify X, it will take it SAT and this is the value of X. So, for that this formula is true and that is the DIP. So, this is how I model the DIP problem.

(Refer Slide Time: 33:41)



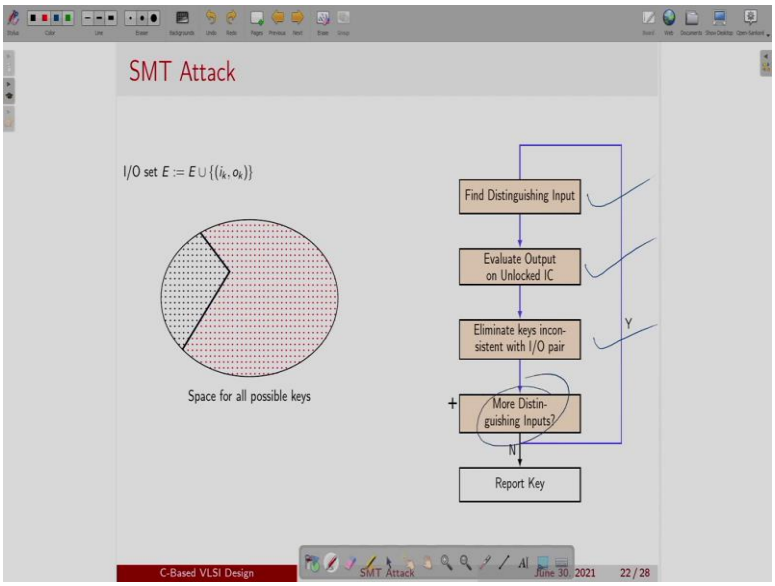
And this is where with some animation, I try to show that so this is my whole search space and whenever you find a DIP, it probably there are two keys which is basically from the wrong set and after that, this part will reduce; not a single value rather it will reduce that whole search space by a small number.

(Refer Slide Time: 34:03)



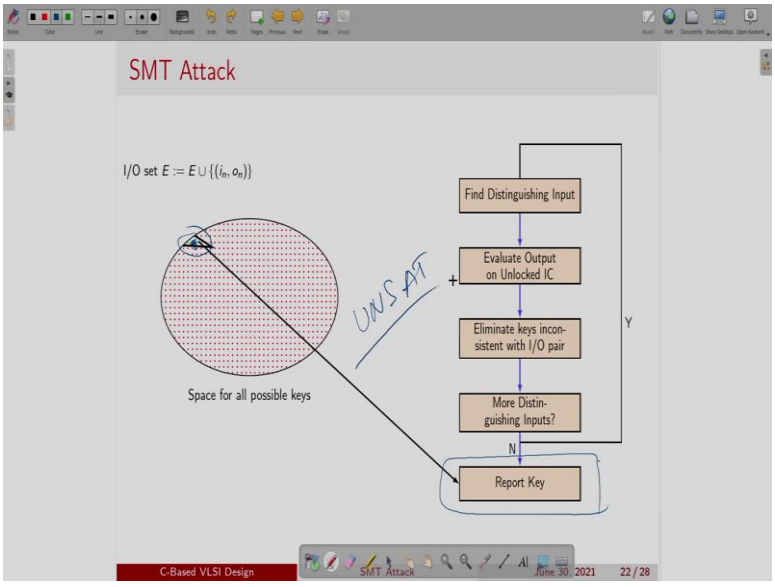
So, earlier this is the whole thing. So, it removes this part.

(Refer Slide Time: 34:12)



And then you, so basically you find a DIP, then you evaluate the oracle, you basically add a constant and it will actually remove a set of a class of wrong keys and this process will go on. So, it will go for find another DIP and so on. So, it will go on and every iteration, it will reduce the search space.

(Refer Slide Time: 34:33)

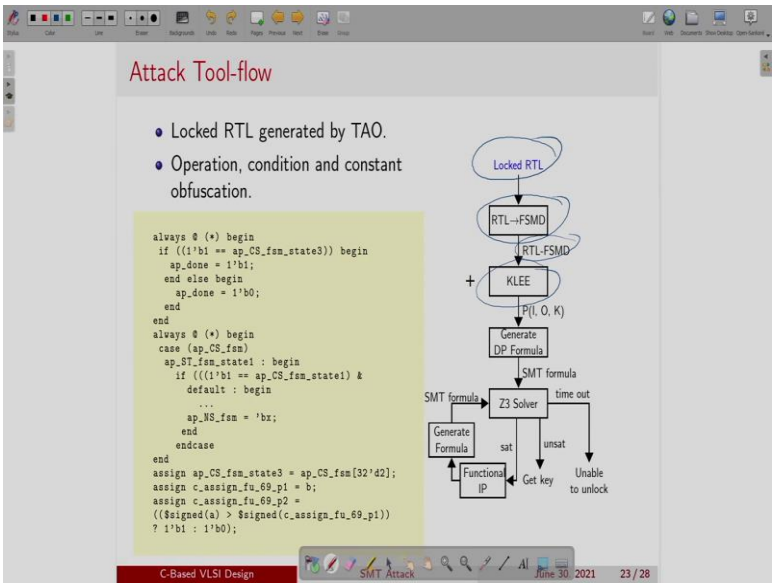


And finally, it will say this is the correct key. So, once say there is no DIP only single key is present, it will it will just say this is the key. So, then at that time, what will happen? It will say no, I do not able to find out any key. So, I will get UNSAT. So, tool is such that that does not exist any DIP for which the output is different. So, it will say

UNSAT. So, then we know that only one key is remaining. So, what I have to do? I have to just go back to the previous formula.

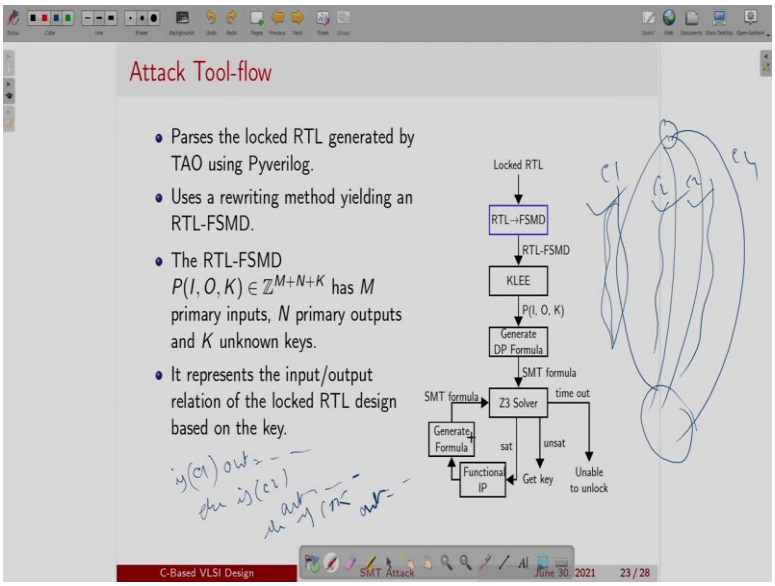
And here instead of output is different, I will just ask you just find a value, where Y2 equal to Y2 because now it does not identify two keys because there is only one key present and say for that output is not different. So, it says UNSAT now if I just change this to this what will say there are same value K1 and K2 is same for that it is happening. So, that way this whole process will go and you will identify the correct key . This is the overall idea.

(Refer Slide Time: 35:41)



So, only thing is remaining to be discuss is that how do we model a program. So, this is the overall tool flow, we model the program symbolically. So, we have the RTL. Using my RTL to C converter, I get an RTL-FSMD.

(Refer Slide Time: 36:02)

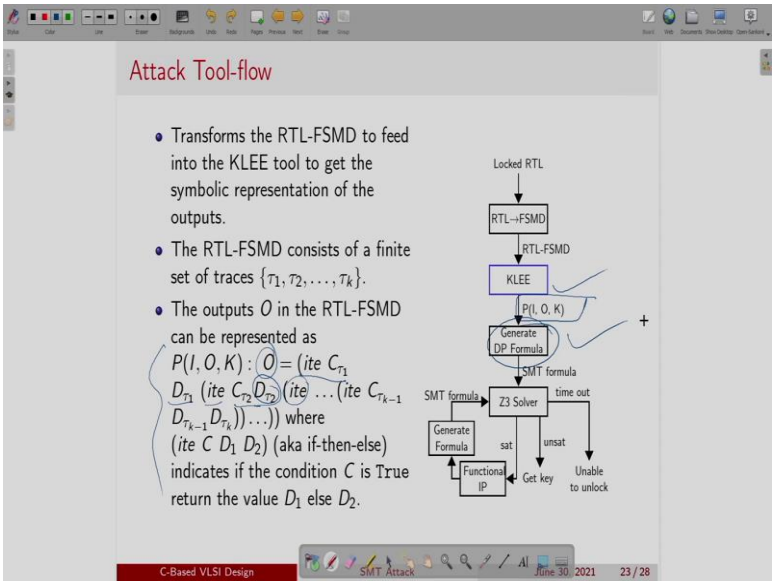


Then, I have used a tool called KLEE, which is basically symbolic simulator tool which will give you the all the paths of your program. So, in a program what is going to happen? There are many paths. So, and each path has a condition and you have set of operation is happening. So, KLEE will identify all the paths in your program and we assume the loop is basically static. So, KLEE will unroll the loop.

So, there is no loop at all. If it is a static loop, you can actually unroll and you will get an unrolled loop. So, then how will you model the program? So, the way I am going to model the program is very simple that if say this is the condition C1, this is the condition C 2, if this is the condition C3 this is C4. If C1, then my output equal to something this expression whatever the expression is there; else if C2, my output will be this expression whatever that comes in; else if the condition is C3, my output is whatever is happening here.

So, this I am going to model this all path in if else; if the condition is C1, then output is this; if condition else, if the condition is C2 and the output is this and so on. So, this is how I am going to model the program. So, that C whatever I just told you earlier.

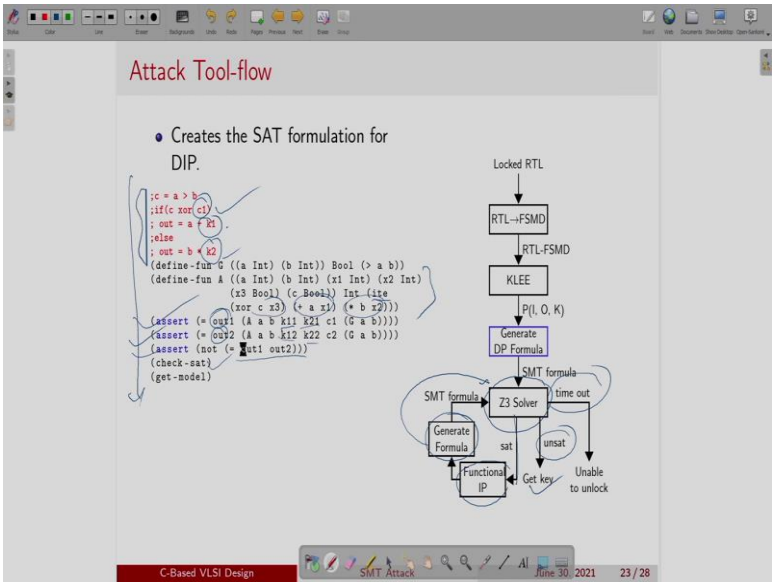
(Refer Slide Time: 37:26)



So, this is what is exactly here. So, my output is depending on that if else So, if condition is C1, my output is D1; else if the condition is C2, my output is D2, D tau 2; else if the condition is C3 and so on. So, this is how I model my program. So, all the path as if else. So, this is what I did here and I have a formula now. So, the formula is basically nothing but a representing my FSMD which is basically a program or a you can say circuit which is this is the program which is extracted from a circuit.

So, this is the formula, then we will construct the DIP. So, basically, we will create a two copy of the formula that and I will just rename that key with K1 and K2 in two different and output is Y1 and Y2 and I just create the DIP formula that I have already discussed.

(Refer Slide Time: 38:23)



Then, I am going to give the things to the SMT solver. So, we have used tool call Z3 to generate this and then, what will happen? Those SMT will basically identify a DIP and then, I will just evaluate that with the DIP that actually oracle and then, I will get a constraint which is the next formula and we will update that formula, we just add this formula and again I am just going to call this Z3 solver. This is the things whole things going to happen iteratively.

Once say there is no UNSAT, I will try to identify the key and since this is a formula and this is the whole problem is undesirable because of the real and integers, sometime SMT solver may say I do not able to prove whether it is a SAT or UNSAT; it my timed out.

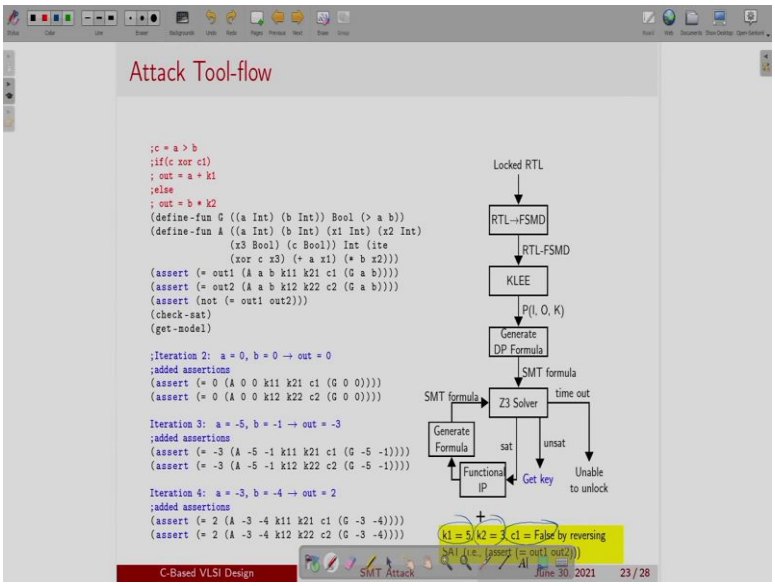
So, that time, I will say we do not know the what is the key and that might happen because of the complex nature of this problem. So, here, I took an example say it is basically if condition is this, then you do this, else do this and here c1 is the key k1 and k2, these are the constant locking is happening.

So, how do I model the program is basically this. You can see here that is basically ite, if the condition is this, then you do a plus 1, else you do this. So, high level is this; no need to go into that. And then this is the DIP formulation. Then, I create two copies.

So, this is the same copy of this formula, where I just use the key k1 and k2 and here this k11 and k12; k21 and k22. So, I just take two copies of the formula because I need two copies and the output is out1 and out2 and here, I am just saying you make sure that output is different.

And then, if you ask the tool key check SAT, it will try to check the satisfactory of this formula. This is exactly my DIP formulation. I have the program model; I have two copies; I am checking the output is different. So, if it is said SAT, it will basically give you an input for which the output is different. So, this is what is going to happen.

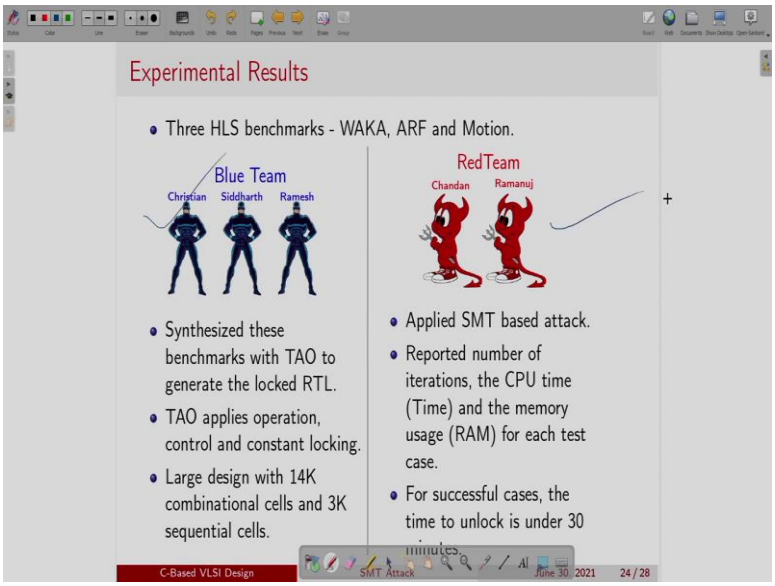
(Refer Slide Time: 40:20)



So, here I took an example. So, say initially first iteration, it just says my DIP is, a equal to 0, b equal to 0 and output is 0. So, with this, I just add these two constraints. So, I will get one constraint because there are two equations here; I have to create two copies of that same thing. It is the same formula, the same constant; but I have to create two copies because otherwise I am not able to constant the other key space.

So, next iteration, it will identify the DIP is, a equal to minus 5 and b equal to minus 1 and the actual output is minus 3 and based on this, I will add the constraint that I have already discussed. This process will go on and finally, after fifth iteration, it says there no further DIP exist and it say that my k1 equal to 5, k2 equal to 3 and c1 is equal to false. So, this is what my correct key for this.

(Refer Slide Time: 41:16)

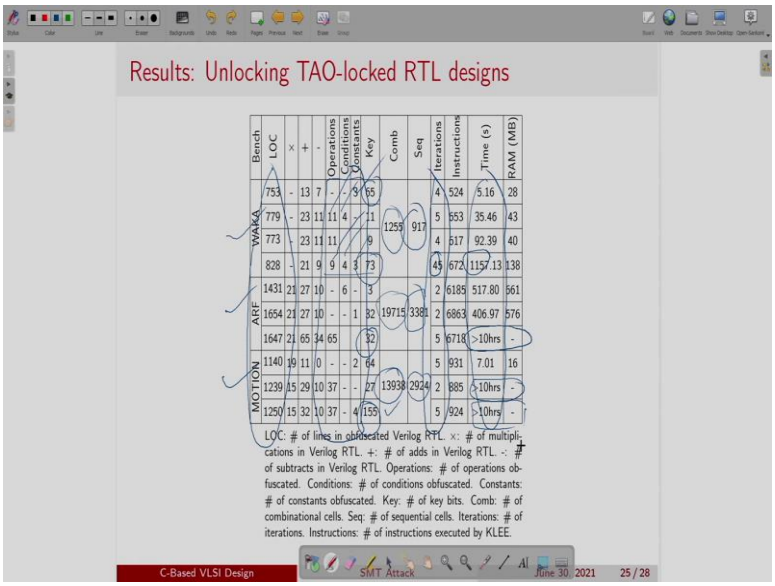


So, this is the overall idea of our tool and that is what we have developed and we have shown that this particular technique can break all the locking techniques of TAO. So, you remember the TAO has applied constant locking, operation locking and condition locking.

So, our this SMT attack can break all these three techniques and we actually construct a red blue kind of team, where we set up people of our group which actually develop this RTL and that this is a locked RTL using TAO and they give me the locked RTL because if I know the key, I can do some trick.

So, the red team which is the attacker team that does not know what is the correct key. So, this team will create the lock circuit and give it to this and this team the red team apply this SMT attack technique and actually identify the keys. So, basically, they develop this and they apply this TAO technique to generate this. This is the overall approach of this red blue team kind of attacks because this is the team which develop the chip and this is the attacker, who will try to identify the keys.

(Refer Slide Time: 42:25)



And here are some interesting results. So, we take three benchmarks; high level synthesis. We can see the number of lines in the RTL is very high. So, it is a big code and here this is the locking. So, this is the locking is applied and the key size is this. So, we can see here 65 keys, 32 keys, 155 keys.

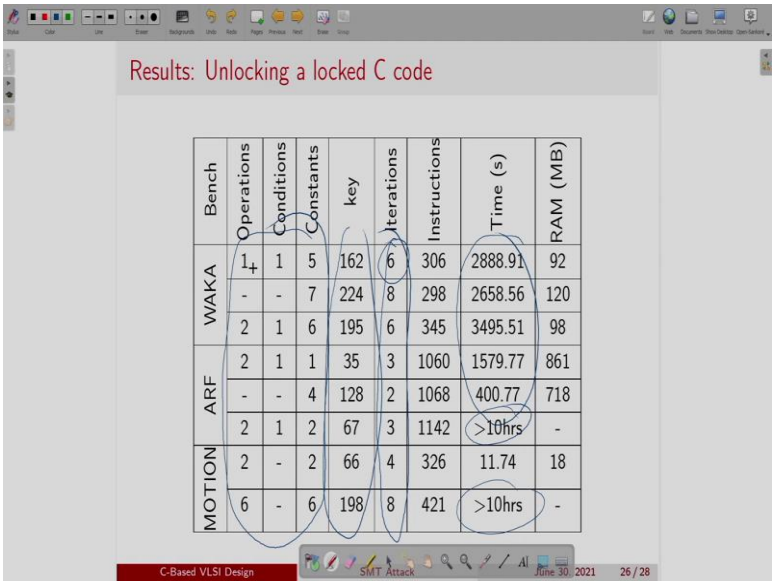
So, we have developed circuit with this many keys and it says that how many constant locking is apply here. Here for example, 11 operation locking is used, 4 condition locking is used; for this case, 9 operation locking, 4 condition locking and 3 constant locking and the key size is 73.

And after doing this, so in the in the gate level circuit, this is the number of gates and this is the number of flip flops, you can see here it is basically a big circuit. So, 13000, 19000 gates and 3000 odd registers. So, it is a big circuit. And interesting you see how many iterations is taking? 4, 5; only one case, it is 45.

But most of the time, it is only 4, 5 iterations and its actually identify the correct key. What is the correct key understood, we just send it to the blue team and they confirm, ye this is the correct key and the run time is basically you can see here 5 seconds, 35 seconds, 92 seconds is very fast. So, everything is within some minutes. So, that is all.

But as I mentioned, the problem of this checking satisfiability of the formula is undecidable. So, some scenario, we are not able to get the key. So, there is 10 hours' time limit; within 10 hours, this tool cannot identify whether the formula is satisfiable or not. So, that is we also have that scenario.

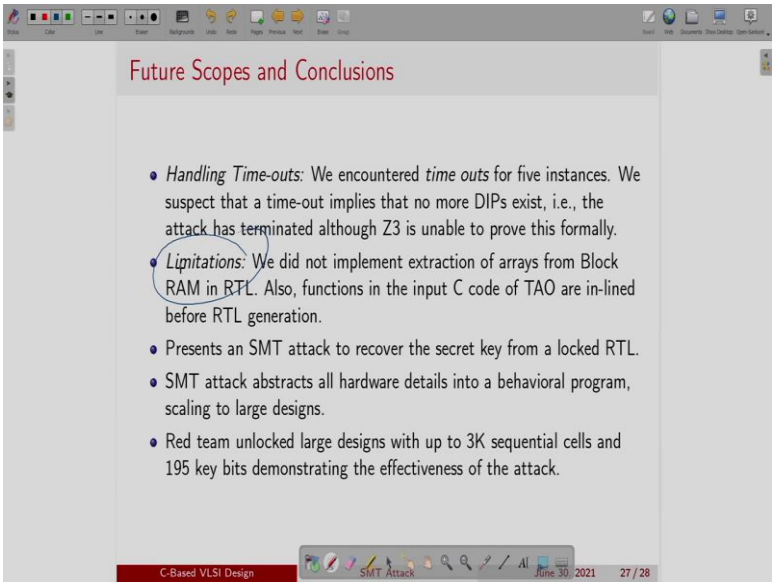
(Refer Slide Time: 44:12)



Then, we create some bigger test case. The key size is 162 bit, 200, more than 200 bits and so on and we run the same process and we are able to identify the keys again and you can see the number of iterations is really low. So, that is the beauty of the DIP. So, even if it is the 2 the power 162 key space; in 6 iterations, we can identify the keys. It is not 2 to the power 162, its only 6.

So, that is the number of iterations you needed to identify the keys and we can see most of the time, our run time is within less than 1 hour and 2 scenarios, where we are unable to get the keys. So, but most of the time, we are able to get the key which is very encouraging and this talked about the which locking has been used to create these keys.

(Refer Slide Time: 44:59)



So, to summarize, we in this particular class, we discussed about a SMT based attack to identify the keys which is locked by a technique called TAO, during high level synthesis and this we have shown that this SMT attack is basically very powerful in identifying the keys because applies a very unique idea of DIP and this DIP helps in identify this key in very few numbers of iterations.

There are some drawbacks also because we found that many time the tool timed out and specifically, we understood that because of the multipliers and this the checking the satisfactory for multiplier is a difficult problem for SMT solver and we identify mostly, it is because of multiplier it sometime gets stuck. So, we have to enhance this overall process how to handle this overcome these limitations. But the overall idea is it is a presentation of a SMT base attack on RTL locking. So, with this, I conclude today's class.

Thank you.