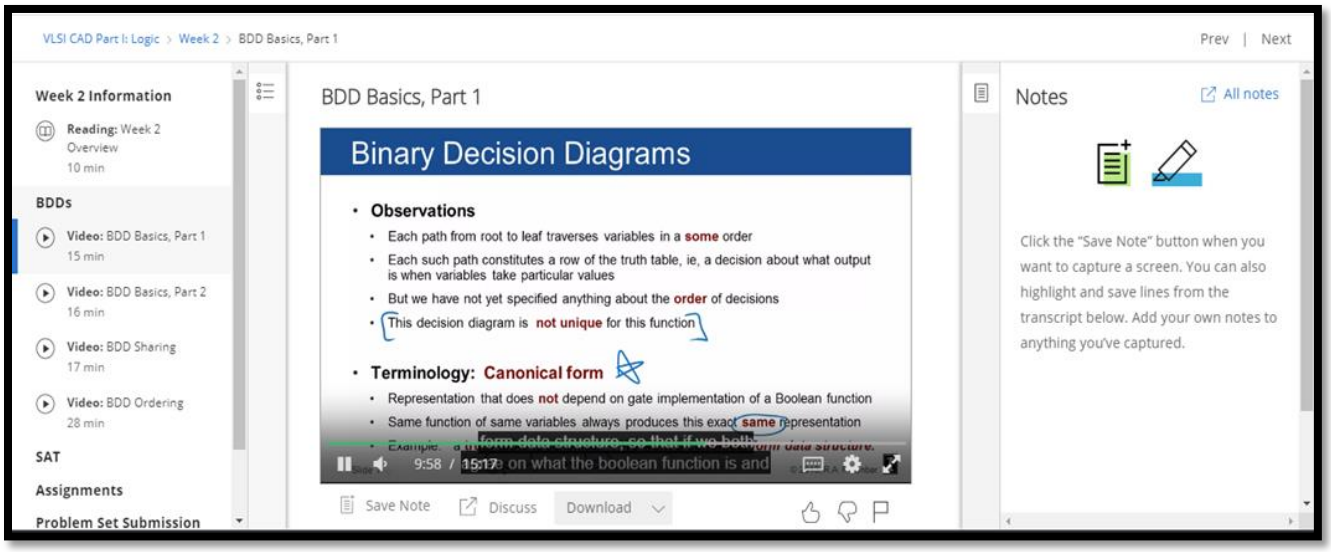
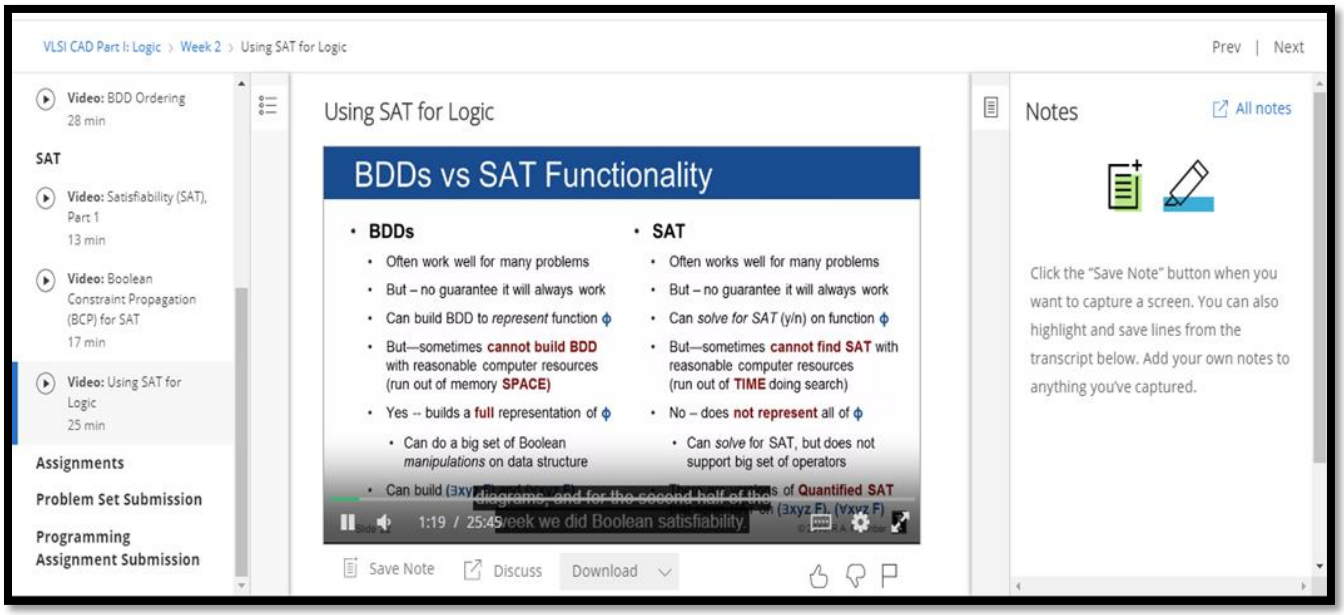
DAILY ASSESSMENT REPORT

Date:	04-August-2020	Name:	Swastik R Gowda
Course:	VLSI CAD Part-1	USN:	4AL17EC091
Topic:	Week 2	Semester & Section:	6th Sem 'B' Sec
Github Repository:	swastik-gowda		

FORENOON SESSION DETAILS

Image of session





Report – Report can be typed or hand written for up to two pages.

BDD Basics, Part 1:

- When we talked about the Unate Recursive Paradigm at the end of the last lecture, we introduced a simple kind of representation and just one basic operation, Tautology.
- Urp is a historically important representational scheme, but it's not really the way we do things to day.
- So what I want to start talking about is one of the most powerful data structures in the Boolean universe and that's the binary decision diagram. So they're called BDDs.
- And in this lecture, we're going to introduce what a decision diagram is and that we're going to start manipulating it and restricting it and constraining it so that it becomes a really powerful data structure on which we can do important operations.
- So, here we are again, focusing on representations and how important they are in our study of computational Boolean algebra.
- Where the goal is to be able to manipulate Boolean objects as data structures that are operated on by operators that we can build in software.
- And as way of example, I'm just showing what we ended up with at the end of the last lecture which was the URP tautology algorithm. And you know this is really a great warm up example.
- We've got an internal node in a tautology computation that has cubes b, c, and b bar c bar. Do the co factoring we see on the left hand side, oh, I can tell it’s a tautology because I've got an [UNKNOWN] cube, there's a one in there. Now on the right hand side we can again see it's a tautology because we've got two unit, two single variable cubes c and c bar.

BDD Basics, Part 2:

- A decision diagram was a way of representing something like a truth table, but just way too big.
- And one of the first things that we did was to decide that we should order the variables in an attempt to make the diagrams all the same for a given piece of, of Boolean logic.
- We're going to continue constraining and constricting the decision diagrams. We're going to introduce another big idea in this lecture which is that, we're going to reduce the diagrams.
- We're going to remove certain forms of redundancy from the diagrams. So that we have a really wonderful property, which is that if you give me a Boolean equation.
- And you constrain the order in which the variables appear in the BDD, I always get the same diagram. I always get the same graph.
- It turns out, that by making the data structure canonical in this way, becomes an amazingly powerful tool for actually manipulating Boolean objects and answering really interesting questions.
- So let's see how that works. We are back again, looking at binary decision diagrams and our first big two big ideas just by way of review, were. The first idea was hey, let’s use a decision diagram.
- And the second big idea was hey let's impose a global variable ordering so that every path from the root to the leaf visits the variables in the same order and what we got was.
- The fact that the BDDs we create are still not canonical. We can have two different BDDs representing the same function.

Boolean Constraint Propagation (BCP) for SAT:

- But more importantly, there's a very powerful and important iterative sort of computation that says, more or less, let's just assume that a particular variable has a value, and let's just see what happens.
- When you have a Boolean equation with many, many variables and many, many, many clauses, it turns out you can get most of the work done with this simple procedure.
- It had a decision part where we more or less arbitrarily decide to set some values to variables. And we go simplify the CNF form and we see what happens. Can we satisfy it or are we conflicted or what happens? The problem is that's only going to go so far.
- In fact, this is where really all of the deep work happens in Boolean satisfiability. And this means stuff has a name.
- It's called Boolean constraint propagation. So, given a set of fixed variable assignments, what else can you deduce about the necessary assignments to other variables? Well, you can do that by propagating constraints. Now, there is a very famous strategy for this, called the Unit Clause Rule, and gosh, this is like 50 years old.
- A clause is said to be unit if it has exactly one unassigned literal, right? And so a unit clause has exactly one way that you can satisfy it and so what that means is that, you have to pick the polarity that makes the clause 1.
- So here is a little Boolean function phi, a plus c, b plus c, not a plus not b plus not c and let us assume that we have decided to make these choices. So a is a 1 and b is a 1.

Using SAT for Logic

- It is just in the nature of solving things that are exponentially hard, that sometimes your heuristic techniques might not work. So, for a BDD, basically, what we can do is we might be able to, we hope, build a BDD to represent the function. And for the SAT side, we can solve for satisfiability with a yes or no answer on that function.
- You want to do a sort of an arbitrary kind of a computational thing on a Boolean function. You probably want to start with a BDD because you just have this rich pallet of operators.
- But, if you just want to solve for the Boolean functions, say, hey, can I make it 0? Yes or no, that's when you want to use SAT.
- So, for example, you can build the functions that represent existential and universal quantifications.
- That's, you know, one of the kinds of manipulations you can do in a BDD universe. In the SAT universe, it turns out that there are actually different versions of SAT solvers.
- Not every SAT solver, but there are versions of SAT solvers. They are called Quantified SAT solvers, where you can say, okay, look here's the function and you give it to him in a CNF form.