Digital System Design
Professor. Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology, Ropar
Lecture No. 61
Register Transfer Level Design

Hello everyone, today we are going to discuss about one more, higher level of abstraction of design which is called RTL design.

(Refer Slide Time: 00:27)

RTL means a Register Transfer Level Design. The basic property or basic characteristic of an RTL design is that input is a read from a register and output is also written to registers. What does this mean? This means that whatever operations, whatever combinational circuits are there, they will read. Their inputs will read from a register and their output will also get written to a register. So, there is some exception there. Me sometimes it is directly or indirectly. So, indirectly or what kind of exceptions could be there.

So, for example, there could be sometime chaining of operations. So, for example, there is a AND gate which is reading from resistor and this output of AND gate is written to OR gate and that output OR gate is written to a register. So, in this case the second OR gate is reading from a AND gate directly, but indirectly it is also reading from the register. So, this is a simpler example.

There could be more sophisticated examples like there could be multiple additions, which are performed in a single cycle. That means, input of input is read from a register by a first adder, but the second, third, fourth adder are taking the output in taking the input from the previous

adders, but finally, the output will get written to a register. So, this is the basic characteristic of a register transfer level design that the operations would be performedwithin two registers.

So, what kind of operations could we performed? These operations could be logical operations like AND gate, OR gate. This could be array of AND gates or a mixture of AND and OR gates. So, which also means that all the encoders, decoders multiplexers all of those operations are possible. Along with that arithmeticoperations are also possible that means, addition subtraction comparison multiplication division.

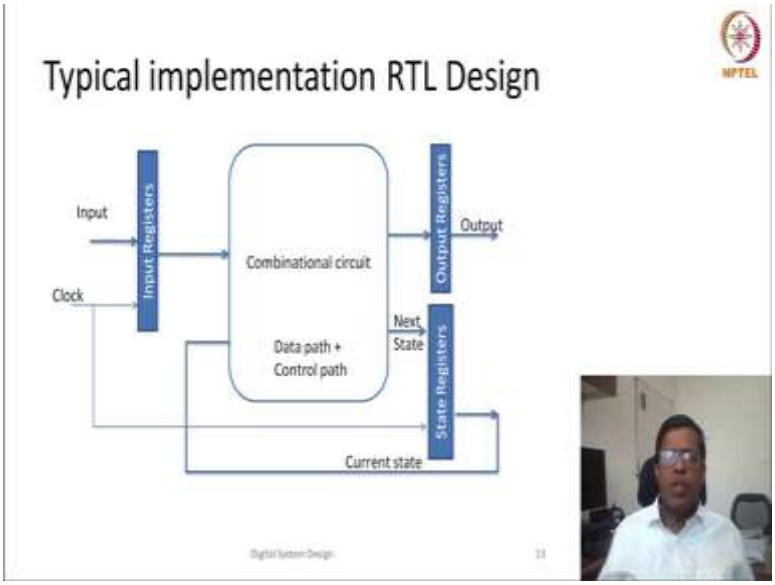
And one more type of operations like memory read, memory write. So, although, memory we have classified as a mix of combinational and sequential design, so, but when we are reading from memory, it is still treated like acombinational design, because data is stored there. So, we are using our multiplexers and decoders to go to that application go to that location and read the data.

Now, these operations could be. So, when we are saying that they are they are read from this registers in written to a register, so for example, if we are writing like this R1 is assigned to R1 plus 1 so that means, that this R1 plus 1 is the operation which is being performed and the input is taken from R1 register, output is also written to R1 register, but because of this assignment operator, the output would be available at R1 register only after active edge of the clock cycle.

So, this is the beautiful nature or this is a different characteristic nature of an RTL design that although, such operations are defined, but the latency is fixed that this operation would be completed within one clock period because all the registers are driven by one global clock cycle. So, because all of them are driven by global clock cycle andthey are assumed to be made of flip flops means that they are they are alwaysedge triggered registered.

So, that means whenever an active edge will come at only after that results would be applicable on the output registers, but during the, so before that active edge the operations can be performed.

(Refer Slide Time: 04:43)



So, this we have this diagram we have seen multiple times. So, the overall discussion we can say that, the input is read from register and there could be a sequence of combinational design. In this combinational design there could be some other gates which are giving output to the other gates etcetera, but overall, this whole combinational design will perform the operation within one clock period, so that the output would be at level after one clock cycle.

So, this and there could be a possibility of multiple such operations happening over different clock cycles. So, that means, in this overall implementation where input is read from the output is written to a register, but there would be a state machine, which is trying to manage the operations and which is deciding that which operation to be performed in which particular cycle. So, these state machines we have already discussed in our previous modules as well as the current module.

So, these state machines will actually would consist of certain type of registers and they we can call them fate registers. So, these state registers would determine that what would be the current state what is the next state. And based on the based on the current state and next state there would be, they would also have their own control logic there would be the logic which is deciding that what would be the next state. So, that logic will also be part of this combinational circuit.

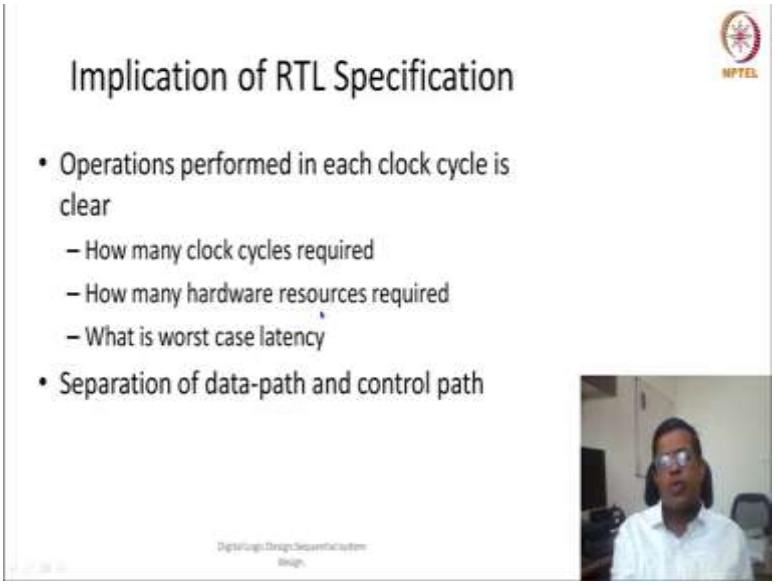
Now, we usually try to disambiguate between these two different kinds of logics. So, one logic which is where most of the logic is, most of the combinational circuit or most of the part of the circuit is actually reading from the register and doing some computation doing some operations, arithmetic operation, logical operation or some other operations. So, and

after doing these operations, they are writing to these registers, so, that wherever this data operations are happening, we call it data path.

So, wherever data this particular data, how that data is being triggered and then transferred and then copied to othercombinational units and then finally, get getting written to output register. So, this overall structure overall logic is called data path. Now, because this data path is usually a common data path for whole the application, so, there would be some control logic which is controlling that when to read which particular register or when to activate a particular operation.

So, all those things then would be decided by control path and this control path is also sequencing of the operations using state machines or using state machines, and state machines would be implemented using these state registers. So, that particular part of the logic would be called control path. So, control path is essentially the logic which is controlling the data path operations which is helping them in selection of data path operations in a particular cycle or in a particular state. So, maybe these things will get clarified when we will go in our, when we will try to do couple of these examples.

(Refer Slide Time: 08:32)



So, in this registered transfer level design, the overall implication is, let us say if a design is specified using RTL level, register transfer level, so, what does it mean? It essentially means that we clearly know that, which in each cycle, which all operations would be performed. So, because we are saying that we know the operations are read from a register and are written to the register so that means, during these two registers, whatever operations are being performed, these, these operations are not.

So, the other implication of this is that the other way of looking at this is that we know that which all operations would be performed in each clock cycle. What does this mean? This mean that if we know which all operations are performed in each clock cycle, so, that means that the overall application would also have been divided and segregated into different clock cycles. And we also know that the overall application how many clock cycles would be consumed, how many clock cycles are required to computeor to do a particular application or to do a particular operation.

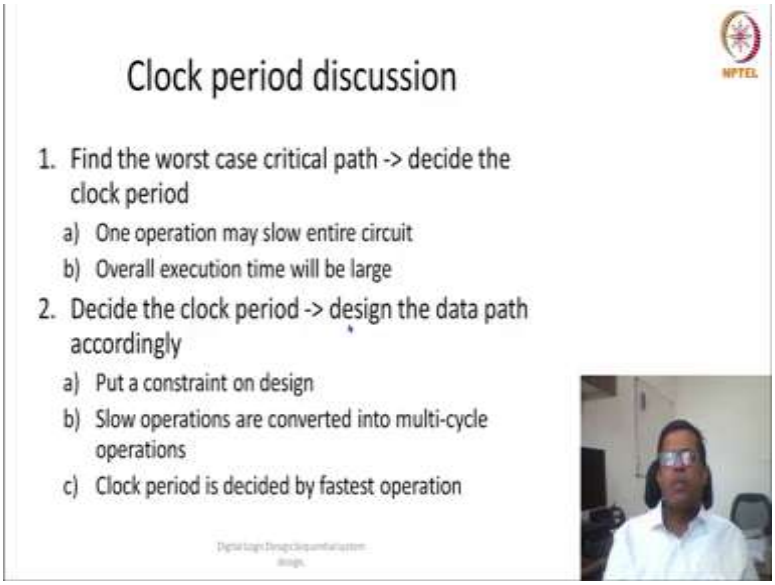
So, sometimes these number of cycles depends on the data that which data, which, what are the values in that particular registers or those things. But if those values are none, then the overall number of clock cycles can be computed. Now, not only clock cycles, but because we know what all operations. So, that means, whatever hardware is required to do those operations that hardware is also known, those hardware resources are also known. So, for example, if the arithmetic operations are adders, multipliers, addition, multiplication, comparison, then we clearly know that how many adder, how many multipliers, how many competitors would be required.

So, how many multiplexers would be required, how many decoders would be required, how many AND, OR gates etcetera, all of those combinational logic is also known, if we know that what all operations are performed within one clock period. So, that also has an interesting implication that if we know all the resources, if we know all the hardware components, that also means that we also know what is the worst case latency. What is the between these two registers between any two registers, what is the worst case delay, what is the maximum delay from one register to the another register is also can be computed at RTL level of design.

So, this also we have already discussed in previous slide that because in the RTL specification it is usually centered towards the data path or the operations and there they are scheduling that which operations would be performed in which particular cycle. So, if those things are known, then we can also have schedule, we also have a state machines which would drive, which will drive these data path, which will say that, which will activate those operations in only that particular clock cycle.

So, that logic that willactivate those operations would be called control path and the data path could be between the registers which is operated by mainly by the data.

(Refer Slide Time: 12:32)



So, here at this point, because we are talking about worst case latency, we are – we have the knowledge of all the hardware resources required. So, we would like to bring this discussion of clock period once again. So, although we have discussed it multiple times during the course, and we have almost converged that a clock period is essentially equal to combinational delay plus t setup time plus t register time or register propagation time.

Now, is equal to clock period. So, that means, in other words, we can also say that ifclock period is known to us, then we are constrained by combinational delay and if combinational delay is known towards that which would, using which we can calculate the clock period. So, these are the two methods I am writing here. So, basically, if I want to know the clock period, one possibility is that I will calculate or I will find out all the possible paths from one register to another register.

So, in other words, I will find all the parts from the register read to the writing of any other register.Now, both the registers are driven by clock, sothey would also be having this specification of setup time and propagation time. So we will consider those calculations also. So, that means thatthe combinational path, which is the worst case combinational path will decide the clock period.

The other possibility could be that we decide the clock period ahead and then we design the data path accordingly. So, to make sure that critical path all the paths are within the critical all the paths are within the length of clock period. So, even the worst case path will also be lesser than the clock period, so these are the two possibilities to say.

Now there is an interesting trade-off between these two. The first one looks quite intuitive that the way we have been designing so far, we will go ahead and design, and after designing we will find out what is the clock period. So, if that is the case, what would happen? So, one thing would happen that let us say incidentally, maybe deliberately or non-deliberately somehow one operation is very, very slow.

Let us say we have some sort of division operation or some operation which is, let us say there is one multiplication operation, that multiplication operation is a combination operation. And we have seen that this multiplication is usually quite slower than the addition. Now, if multiplication and addition itself is much slower than multiplexing, demultiplexing, decoding, encoding and all of the logical operations.

Now, if and it is quite possible that multiplication is a rare operation not a regular operation. Now, if multiplication is a rare operation, then what may happen that due to this multiplication everything will get slowed because my clock period is determined by the multiplication.

So, let us say, incidentally, let us take an example that my multiplication is 10 times slower than my addition and addition is also 10 times slower than shift operation or other logical operation like multiplexing etcetera. So, that means, in a particular cycle where we are doing only multiplexing or we are only doing shift operation, they are 1000 times faster than the clock. So, that means, during that clock period only 100 of the time some operation is happening, otherwise, all other times, it is idle.

So, but we said in our previous slide, so, the overall execution time is determined in terms of number of clock cycles, number of clock periods. So, that essentially means that if one operation is slower and that particular slow operation become a critical path and that critical path is the deciding factor for my clock period, so that means, I have to keep a large clock period. And if the clock period is large, so, that means, overall execution time of that particular task would be quite large.

Now, see one more aspect of this that usually the clock is going to be a global clock So, global clock means there could be different kinds of operationsin my design, so, there could be different sub modules. If this global clock is going to decide the, this global clock is going to be going to decide all the operations on my chip or on my application so, that means, this worst case has to be the slowest operation in among all the operations in the application.

So, in that case it could be even worse because the slow operations like traditionally division and any floating point operations are considered to be much slower than even addition and multiplication. So, in worst case this could impact my clock period, as well as the overall execution time quite badly. And even if we take the safest case, where most of the operations are of similar time, but there are some operations which are let us say 10 percent, 20 percent takes more time takes there delays is little more so the clock period will always be determined using those operations.

Then also, if we are determining our clock period based on the critical path, worst case critical path then execution time is always bound to be larger, than the fastest execution time which is possible. So, that means this does not look like a quite a prominent quite a good option that we first decide the critical path and then decide the clock period. So, one more aspect is this that this also means that we have to design we should know RTL of all the components of the chip.

So, if we can consider some production chips, wheredifferent components are designed by different teams, and all these teams are sitting at different parts of globe. So, let us say one part is done by Indian engineers, one part is done by American engineers. One part is done by German or European engineer. So, because their specification could be different their style of modelling could be different. So, unless they will all converge, and they will tell this is the clock period then only the execution, then only design could start or design of one team could impact the design of the other.

So, because of all of these reasons, this particular method is usually not considered, but the other method where we decide the clock period and then design the data path. This is this is more prominently used in in design industry. So, what would happen if we are deciding the clock period so that means, we are putting a constraint on the design itself. So, far the only constraint or design we have kept is we thought that it is only the area or it should be the delay, but here we are putting an additional constraint that given a clock period now designed the system.

Now, a couple of questions are coming intuitively to my mind. So, what would happen if my combinational delay is more than the clock which has been decided by somebody else? So, now, this clock period has been dictatedby somebody else I will say part of specification that now, we have to design a system with let us say 1 gigahertz, 2 gigahertz, 2.5 gigahertz. So, what would happen if my combinational delay does not fit into this clock period?

So, then we have two options; either we have to redesign our combinational path, so that it gets fits into the clock period. And that is the reason we have when we were designing adders or multipliers we have seen so much of different variations. That if a need would arise that we have to make our adder so fast that it has to fit into a particular clock period, so we can choose a particular design point. Similarly for any other unit therefore, there are different options that we can design it in a faster manner.

The other possibility is there that even after doing all those or tricks, all those design tricks, we are still not able to compress our design, our unit within that clock period, then we have to convert that particular unit into multiple cycle operations. So, we will say that, that particular that particular operation will take let us say 2 cycles, 3 cycles or 10, cycles, whatever. So, that is how it is done.

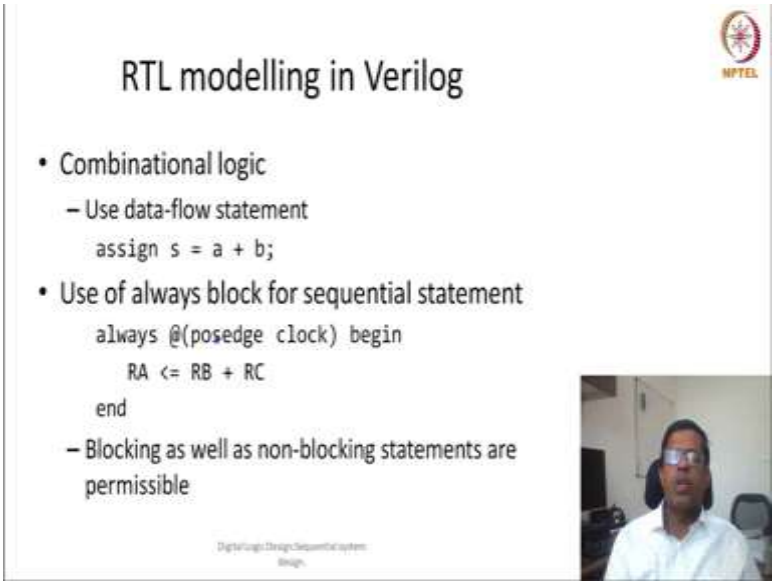
And then still, you may ask that, who is finally deciding? What is the clock period? So, although, you are saying that it is a part of specification, but somebody would be deciding that specification, and there must be some rational behind it. So, usually the rational is what is the fastest operation or what is the most common operation, which is also quite fast plus, you add the register delay, you add the setup time of a register, and then you say that this is the clock period.

So, in many of the cases, typically, it is not logical operation, but arithmetic operation, which decides the clock period and arithmetic operations which are quite common. So, for example, addition, subtraction, all of these operations can be done by our arithmetic logic unit, which can perform addition subtraction, shifting or some time multiplication is also include included, so, it is decided by the whether they are the common operations or not. So, if they are the most common operation, then they will decide what is the clock period.

So, it is also. So, in other words, usually my addition will decide, in many other cases my addition will decide what would be the clock period. So, addition time plus my t set of time plus t resistor time or t propagation delay. So, and because of that, it is quite possible that multiplication takes multiple cycles or division takes 10s of cycles and your floating point operations may take 20s cycles. So, all of those operations would be decided by what is the clock period as decided by the fastest operation, and also the common operation.

Now with this, let us, this gives us some understanding that how clock period is decided and what are the different trade-offs.

(Refer Slide Time: 24:53)



So, if we have to design this kind of RTL model in Verilog, so the way we have been designing so far that we are using the data flow statements, using structural modeling, that is all the way of RTL modelling. So, that is the all the combinational path can be designed or can be modeled using data flow statement or structural statements. Now, if we have to, we also have to do some sequencing, we also have to do these data, register assignment operations. So, all of those operations are specified by at the rate, always at the rate post, post edge or basically the active edge of the clock and then it is written.

So, this particular statement, if you remember, it is called a non-blocking statement. So, you may ask that is it only non-blocking statements, which are allowed in Verilog if we want to model at RTL level then no. Blocking as well as non-blocking statements both are equally permissible.

While modelling while doing RTL modelling, we also have to make sure that the sequencing of the operations is also well defined. Ether it is defined using this always block or you have to design a separate state machine and that state machine has to be implemented in some part of the sequential circuitor another like there could be another always block which is just calculating that when operations scheduling is happening or when a particular operation can be executed. So this is the overall overview of RTL modelling.

Now, to understand it further, let us try to understand it with one example.