

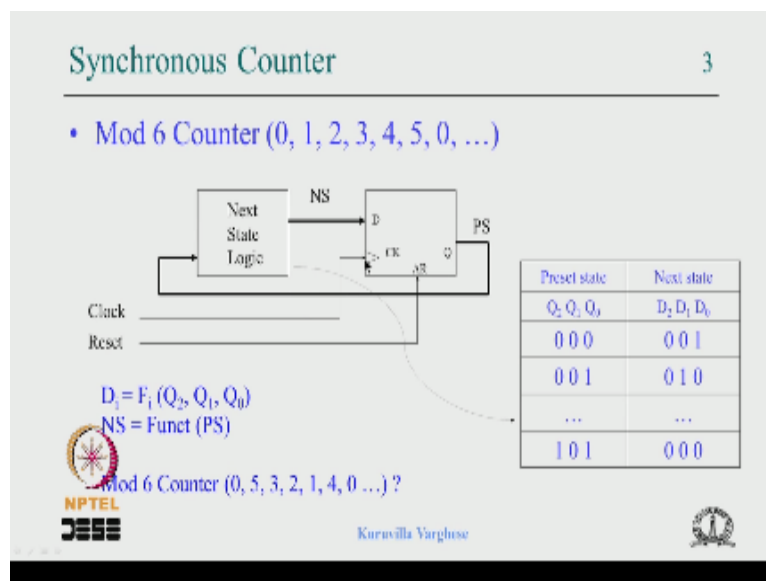
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science - Bangalore

Lecture-05
Top-down Design

Welcome to this lecture on advanced digital system design in the course digital systems design with PLDs and FPGA. In the last few lectures we were looking at the structure of the sequential circuit, how to design it and in particular the last lecture we had a look at the basically the timing analysis, we have looked at the maximum frequency of operation hold time violation.

Then we have looked at setup time when there are setup and hold time in the rescues in the data path and the clock path. Before moving to today's part we will have a look at quickly look at the previous slide, so that period continuity.

(Refer Slide Time: 01:15)



So moving to the slider we were talking about synchronous counter and we said that designing a synchronous counter means putting the flip flops all clock by the same clock and decode at the next state from the present state and that is the logic and we form a truth table and then we get right the next state as a function of present state and that is how the design is done any counter can be designed.

(Refer Slide Time: 01:46)

Output Waveform

4



- $3 t_{co}$, for Q_2, Q_1, Q_0 , Worst case is taken



Kuruvilla Varghese

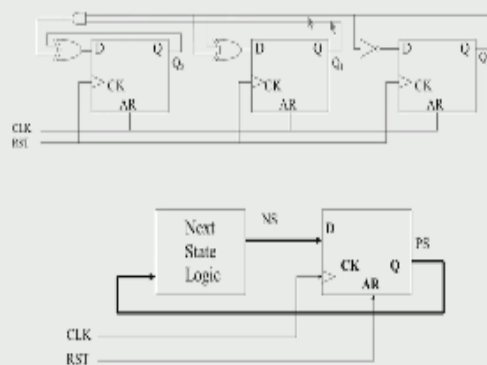


And with when the clock comes with the delay of PCO the present state of the count changes and since in this case there are 3 output will take the worst case as the maximum delay as the TCO.

(Refer Slide Time: 02:05)

Detailed and Next Level

5

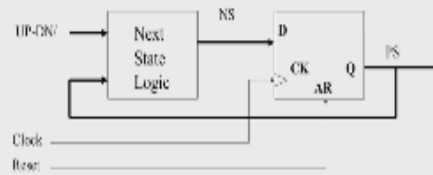


Kuruvilla Varghese



And I mention that definitely there is when you show the picture in an abstract form the details which is not forget, that here there are different path is not just 3 parts from Q_2, Q_1, Q_0 that could be path to D_1 to D_0 . In this case there is only be D_0 as only 1 path, D_1 as 2 path and D_3 as 3 parts from the output. But in general case you know there could be there are end beds of the could be N square paths in this case.

(Refer Slide Time: 02:43)



Kunavilla Varghese



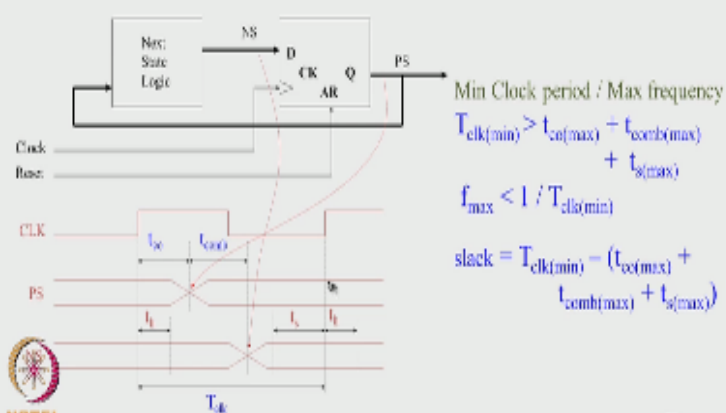
And we also looked at and uptown control essentially we are modifying the logic, we are giving this is an Input and forming a truth table and coming out with equation for the eyes and in this case we find the next it is a function of person state and imports and the next question we have looked at is that can be kind of get rid of this flip flop.

And make it a synchronous it is possible but then you can have that can be reached output research which is difficult to control but asynchronous circuit is very fast because there's nothing like them o clock I know kind of blocking it an opening the path you know to control it, so it can be very fast, it can move another maximus speed but then it is difficult to control because of the unbalance path delays.

(Refer Slide Time: 03:44)

Maximum Frequency

10



Kunavilla Varghese



And this is the essential thing we have looked at the last class in this case for a counter there are two timing issue we have to look at what is the maximum frequency. So the minimum clock period is the delay of the flip flop TCQ, the combination delay plus setup time. So that we are analysing from one clock head to the next clock head and so here from 1 to the next.

So at this point you see the data comes after the delay TCO, at this point it is coming after $TCO + T_{com}$ and we know that data has to be set up as setup time before the next clock head, so the minimum clock period is $TCO + T_{com} + T_{setup}$ and we give a margin cause slack and other the maximum frequency is inverse of the minimum clock period, and one surprising thing is that the whole time does not picture in the whole hand expression for the clock period.

But we have to make sure that the whole time is not violated, that means that data this point should remain there sometime after the clock head and if you just think about it it is enough if you think like when the clock at comes how long it is going to it takes to change this next sheet that we know that if a clock head comes the it takes $TCQ + T_{com}$ time for this to be changed and that is shown here.

So it is enough that the minimum $TCO + T_{com}$ is greater than this whole time then there is no violation ok. So that is what is mentioned here $T_{co\ min}$ and $T_{com\ min}$ is greater than the $T_{whole\ max}$. In this case we take care of it and we have to consider the maximum delay but in this case it is the minimum delay and if you find that you know suppose you fix in a design the clock period and if then the worst case delays delay path violate that clock period.

Then you have to reduce the clock period okay that means sorry reduce the clock frequency and increase clock period ok, but if there is a whole time violation we find that $T_{co\ min} + T_{com\ min}$ is kind of less than whole time the clock is not something to be considered because in the earlier analysis we are considering the timing from 1 clock to the next clock head.

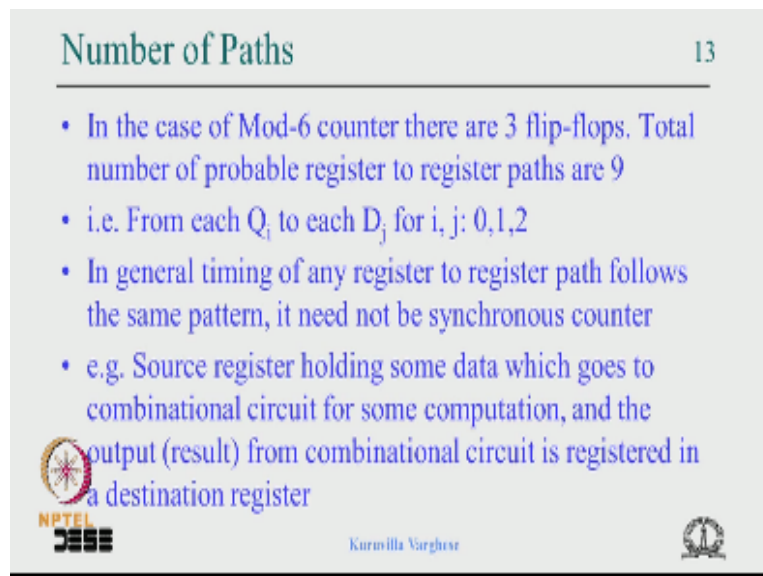
But in the case of whole time violation we are talking about the same clock head ok. So there is only 1 way of kind of solving the hold time violation is to increase a combination delay or I had kind of dummy logic to increase a combination delay and this can happen you know this whole time violation can happen when there is combination delay is minimal or there is no combination delay like in a shift register.

Normally in a flip-flops have their TCO is greater than the whole time, so normally there is no question of violation ok, but like in this case we receive one thing out assumption is at the clock edges I just read all the flip-flops at the same time ok which is not a realistic assumption that could be skew between the clock arrival time at the various flip-flops and that skew can create the whole time violation.

We will analyse these two timing issues with respect to the I mean when the rescue that we will do little later when we probably take the FPGA lectures we will kind of consider that for the time being just they will do the symbols ok, but as I said before at the introductory lecture the basic timing parameters for a combination circuit is the propagation delay and for flip-flop is setup time hold time and TCO.

Now you see that when it comes to sequential circuit with this T_s T_{hold} TCO and T_{com} we are building you know we are going one level up, we are coming out with expression for the minimum clock period and condition for the whole time violation. So these two are the kind of important timing detail of the sequential circuits or any registered data path that has to be understood.

(Refer Slide Time: 08:58)



Number of Paths 13

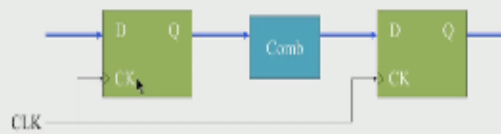
- In the case of Mod-6 counter there are 3 flip-flops. Total number of probable register to register paths are 9
- i.e. From each Q_i to each D_j for $i, j: 0,1,2$
- In general timing of any register to register path follows the same pattern, it need not be synchronous counter
- e.g. Source register holding some data which goes to combinational circuit for some computation, and the output (result) from combinational circuit is registered in a destination register

NPTEL JEE Kuruvilla Varghese

(Refer Slide Time: 09:03)

Register to Register Path

14



$$T_{clk(min)} = t_{co(max)} + t_{comb(max)} + t_{s(max)} + slack$$

$$t_{co(min)} + t_{comb(min)} > t_{h(min)}$$



Kuruvilla Varghese



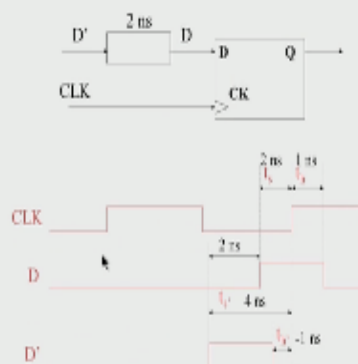
And next thing we have looked that was like it is said this is applicable to any register path you know, after all that picture shows at the data moving from one register through a combinational circuit to another register. So that is with respect to a sequential circuit but in a scenarios where there is we call data path, where there is computation this will be the structure there will be registers of flip flops, combinational circuit and registers.

The same analysis hold good, there is a minimum clock period which consists of TCQ max, Tcom max and T set up max and the hold time violation at the destination register is TCO min+Tcom min should be greater than the whole time ok. So that is a generalized or a data path or register to register part or a sequential circuit ok.

(Refer Slide Time: 09:59)

Setup, Hold Times with skew

15



Kuruvilla Varghese



And we have looked at the setup and hold time with skew when there is a skew in the data path when we refer to that external. The set up timing greases because this delay I mean you have to set up the data much before so much time before at this point ok so you see that with respect to B dash here the setup time is increase to 4 nanosecond, but the whole time has become negative.



Because at this point the whole time spec was 1 nanosecond that means it remains there are 1 nanosecond after the clock head, but since this delay holds a value for 2 nanosecond in principle like you can remove the data with respect to clock like 1 nanosecond before the clock edge at this point and at this point will be correct. So as I said the set up time is measured as a time before the clock head to decide when the whole time is measured as a time after a clock head to that side ok.

So when the whole time is towards the left of the clock then it becomes negative and negative whole time means that at the point of reference it can be removed the data can be removed before the clock head, but this delay will hold it and the other for the input of the flip flop at that be correct ok, so you do not need to worry about a negative whole time.

(Refer Slide Time: 11:42)

Setup, Hold Times with skew 16

- Most often, setup and hold times of flip-flops or registers with respect to a pin or output of another register need to be analyzed.
- When there is a delay t in the path to **D** input, the setup time with respect to new reference point **D'** is increased by t and hold time is decreased by t .
- In this case, hold time can take a negative value. A hold time of $-t$ means that at point **D'**, the data can be removed or changed t time before the active clock edge.
- **Note:** Setup time is defined as time before clock, data has to be setup. So, for setup time positive value is going backward from clock edge, and negative value means it is forward from clock edge. For hold time reverse case applies.

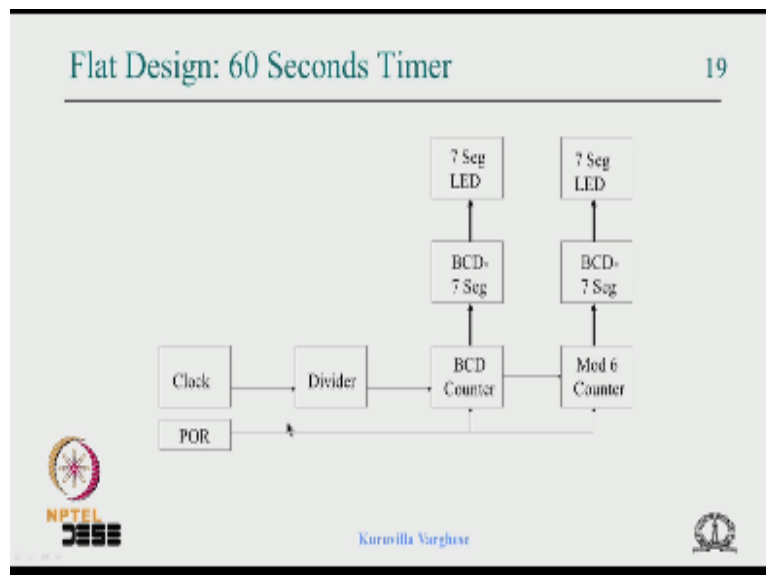

Kurusilla Varghese


And if opposite is a case when there is a delay in and the clock path or there is a skew in the clock path then we have a clock dash which is at this point which is coming earlier to this clock. So now you see that the set up time was 2 nanosecond with respect to original clock at this point and the whole time was 1 nanosecond. Since the clock dash is coming earlier to it.

And we can actually we can and now set up the data much later at this point, because clock is at this point clock comes only after 10 in 3 days in so you can see that the set up time has become negative that means we are setting after the data at this point with respect this clock point after the clock heads, but when it comes here to be correct, but the whole time is increase ok.

It has become 1+3 nanosecond, this has become kind of 2-3 which is -1 nanosecond and so here the whole time increases and the setup time decreases it can even go to negative value as I said I means that you can set the data set up the date after the clock head that is meaning of it.

(Refer Slide Time: 13:02)



And this was the last part of discussed suppose we talked about the design now you are coming to designing the circuit, designing the system and if you take a simple example like a 60 seconds timer, we have seen the design, you know you take clock oscillator that divide it to get a 1 hours like you have a BCD counter counting 0 to 10 that that goes to a mod 6 counter then behaviour BCD to 7 segment Decoder and this goes on and only goes from 0 – 59 0 and so on ok.

And we said it is fairly simple circuit, we have design from one end to other end but we see that are the issues like what should be the top frequency higher the better because if it is higher the drift will be less and for the same drift the high clock frequency since it can divider that will be much better, but we have choose higher clock frequency the divider will occupy

lot of area and we said that you can instead of having a BCD counter followed by Mod 6 counter.

we can have a Mod 6 counter ok, so here there are 4 flip flop, 3 flip flops, in the case of Mod 6 counter you need only 6 flip flop, and that is quite natural because a BCD counter all the flip flops are not used to the full extent because 4 flip flops can you now count up to 16 then only we are using 10 ok, but there is no guarantee that if Mod 6 counter Mod 6 2 7 decoder will have lesser area than.

This put together and we also said that this has to drive LED so it has give high current. In a simple problem there is a question of area the speed and this divider might kind of 10 megahertz the flip flops should work with 10 megahertz, so that is a kind of little high frequency compare it and so there is a timing issue the area, speed, electrical you know the currents and all kinds clock edge.

So accuracy even in a simple circuit there are if you put a mind there are lot of issues to handle, so but this is a flat design given the spec you can design from one into other end.

(Refer Slide Time: 15:42)

The slide is titled "Design Issues" in a teal font at the top left, with the number "20" in the top right corner. Below the title is a horizontal line. The main content is a bulleted list in blue text:

- Accuracy
 - Clock Frequency
- Area
 - Clock frequency, Divider
 - BCD, Mod-6 or Mod-60 counter ?
- Timing
 - Max frequency – Divider
- Electrical Specifications
 - 7 Segment LED driving

At the bottom left, there is a logo for NPTEL (National Programme on Technology Enhanced Learning) with the text "NPTEL" and "JEE" below it. At the bottom center, the name "Kurusilla Varghese" is written in a small blue font. At the bottom right, there is a small circular logo featuring a classical building facade.

So we are going to look at a fairly reasonably complex design.

(Refer Slide Time: 15:49)

- Example 8 bit Micro-processor

- 8 bit ALU, Data Bus $D_7 - D_0$

- 4, 8 bit registers

- 16 Instructions, (4 bit op code, 2 bits each for src & dst Register)

- 64 KB Address space, Address lines $A_{15} - A_0$

- Program counter 16 bit, Stack Pointer 16 bit

- No separate IO space

- Controller – hard wired (Not Micro-coded)



- De-multiplexed Address and Data bus

- Interrupt

Kurusilla Varghese



So let us look at this example let us look at how to design a 8 bit microprocessor definitely I am not going into complete design of the microprocessor, I am trying to take this as an example of a resemble complex design case and illustrate you have go about designing a complex design problem and what are the steps involved and how do you handle it. All that is what is our focus.

I am not guarantee in that in the end of it we will be completely designing a 8 bit CPU, though at the end of the course you can do it nothing very specifically grate there are lot of detail, lot of functional detail, lot of timing detail, but it can be done, now there is no problem and we are also looking at a very simple architecture ok now very efficient architecture.

The main point is illustrating the process of design and for you to understand you know without at the very very used way, so I am awarding all kinds of you know high performance design or simple clock design and things like that. So let us look at the spec of the microprocessor, so it is an 8 bit microprocessor means it means 8 ALU, data bus is 8 bit, D_7, D_0 . It has 4 register, which are of 8 bit, it has 16 instructions.

So the purpose of choosing a 4 registers is there we are hoping that there are 16 instruction which will occupy for that, then you know that in the instruction we have to specify the registers. So 2 bit for codes and 2 bit for destination, so that makes it 8 bit, so there could be most of the instructions could be worn by would most of them, but we have instructions like Jumbo call Vikas definitely specify the address and that can be kind of multiple bytes.

Maybe one 8 bit for opcode and 16 bit for address and so on. It has 64kb address space in A15 to A0. So since the program counter and stack pointer are the one which is giving this address it has to be 16 bit. There is no separate IO space that means the memory space is used to map the IO devices, it is easy otherwise you have to have a separate instruction which is handling the USBs and all that.

Also it is map on the main idea is that you should have enough address space then you can accommodate the memory and IO in a single space you do not need to have a separate IO space as a controller is hardwired that means it is a finite state machine it is not something about micro coded as in very early designs and if you use de-multiplexer address and data bus, separate orders plus there is 1 interrupt ok. That is the specification okay.

So first thing is I have a look at it it's almost clear that we cannot design as we design a 60 seconds countdown. Now you cannot start putting say ok let us put the ALU, ALU you has addition, subtraction, complementary operation, logical operation and comparator, all kinds of thing. So there no way we can just and just put start putting the blocks and interconnecting as we did in the case of that second counter there are registers that need to be designed there a program counter.

(Refer Slide Time: 20:30)

CPU Design 22

- Partition: Functional blocks with interfaces (signals)
- Top-down Design
- At each level
 - Specifications / Functional description
 - Timing specifications
 - Electrical specifications

NPTEL JEE Kuruvilla Varghese

If you are not kind of design at least once you do not even know how to design it ok. So this is a problem which cannot be kind of design from one end to other and in a flat way. So basically in such cases have to bring this of huge design into pieces we have to call it as partition, way to partition the functional block with interfaces that means we should say we

might divide this is ALU, registers, the program counter, the stack pointer and then we say how they are interconnected and so on ok.

So that is the basic idea partition and we go top down that means that we go from the CPU at the top as a single block, then we come down to break into level 1 pieces like ALU, register, the program counter, stack pointer, then we take ALU and further design it which is composed of adder, subtractor, comparator are the logical operations and so on so. That all from the top to bottom we come down as you see come down to the bottom there are lot more blocks, quite a lot of blocks.

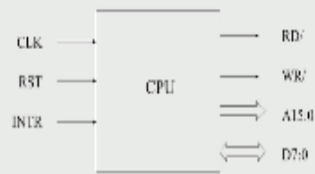
Each block at the level 1 will be kind of exploded in the level 2 blocks and so on. So that is called top down design and at each level we have to look at the functional specification, timing specification, electrical specification and all that ok. Now when I said top down design this is applicable to any design whether it is kind of software design or an aircraft design or organising a complex function all that.

You know not say you up to organise a big problem then you have to have come it is looking at the whole arrangement the speakers, then the food and mementos and so on. You know that there are finance or lot of things, so that is all any complex things has to be broken down into pieces and handle that is all, only thing is that you have to have experience in the game.

Otherwise you cannot do it if you have not organise a big function it is very difficult to organise the first time. So you have to get in a thing, get the experience and do that, okay similarly if you want to do an aircraft design it is composed of various pieces, unless one go through it for a long time get enough experience then only you can design that similarly with dual system when you want designer complex things like process you should know about processor.

And you should know what are the blocks of the processor, how to design them, how to integrate them and all that so you need a very good domain knowledge and experience and expertise to do it, but my point is that the only way to do is that a top-down approach I know going from top block all the way down to small blocks.

(Refer Slide Time: 23:14)



- Functional: Multi-cycle Execution, Instruction set, ...
- Timing: Bus Cycle, Interrupt, Clock, Reset, ...
- Electrical: Bus driving



Kurusilla Varghese



So let us look at the Sheep you at the top level, so in our case are this is the block diagram of the CPU at the top level we have talked reset an interactive input read write an address as output and that has bidirectional you know what and so many people kind of hesitate to draw such a block diagram at the top level it may be simple but nevertheless you have to draw it to bring clarity.

And what are the functions at this level okay, there are definitely there a lot of functional spec have to decide at this point of time, you have to decide whether the CPU is going to execute instruction in a single clock cycle or multiple clock cycle it has to be pipeline what is the instruction format, what type of instructions you support all that comes in kind of functional aspect.

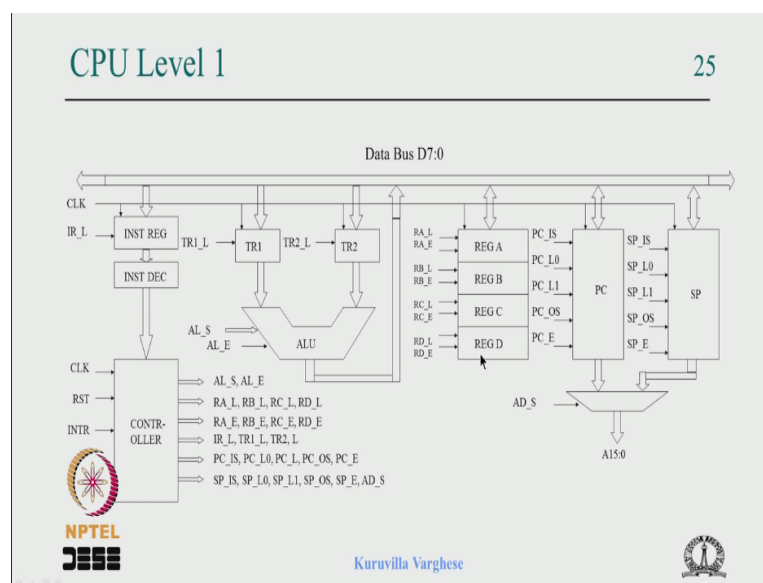
And when it comes to the timings spec you might think there is much there is much timing at this point of time but you see there is definitely there is timing on the clock reset, interrupt, but very important thing is that you know that the memory and peripherals are connected to this data bus be the CPU is along with the peripherals is in a single chip or it is external does not matter.

But you know that there is something called the bus cycles which essentially access memory and peripherals, so it goes like this maybe in the first clock cycle the address come, the second clock cycle the rebug come comes then the data comes on the data bus, so that need to be specified, how many clock cycle in there for the bus cycle to complete for an instruction related to the memory or peripheral to complete.

And that could be asynchronous, there could be synchronous. So there is timing spec and when it comes to electrical specs we know that this address and data lines in these lines are connected to multiple peripherals and memory. So that should have good driving the that that it has to source and sink large amount of current is not in terms of my comp but it has been the order of milliamperes. Then only it can drive multiple loads.

So all these are the spec at the top level, we call level 0 and now we break this into kind of level 1 ok.

(Refer Slide Time: 25:59)



So let us look at the level 1, so I must say that this is a something which I have put it very quickly for the instruction purpose, I do not claim a very great accuracy, there could be some kind of timing issue, but basic idea is sound, you can kind of follow this and make a CPU is possible and this is a very simple CPU which works in multiple cycle that means I take maybe multiple clock cycle for instruction to complete.

And one instruction is stretched it is executed then only the next instruction is refluxed, so it is a multiple cycle CPU. So let us look at the partition for you see that there is an instruction register in Skoda, a controller, ALU with temporary registers, for register a program counter and stack pointer ok. Now the program counter or the stack pointer depending on the inspection of the situation drives address bus.

There is an internal data bus which connects all the data path elements or all the registers and combination circuit basically registers because the combination circuit cannot be directly connected to the data bus only then because there is a single bus only one of the kind of output can drive. So if you have a combinational circuit you should put a kind of tri-state bus to connect to the database ok.

So now so that there is an internal data bus, but the modern CPU might have separate buses because of that many things can happen parallel because there may be a bus for the instruction to get into instruction register, there could be bus between registers and ALU and so on okay. Of course has to be at some point interconnected, but for simplicity we assume a single data bus.

So let us look at these four registers and registers are you know in a 8 bit registers and it gets the data input from the data bus, it also drive the data bus, ok because some time an ALU output has to come to the register or an instruction, part of an instruction might get load into a register, sometime the register has output has to go to the data bus to the ALU input or to the memory and so on or depending on the instruction.

So register, input and output both are tied to the data bus input is not a problem because all the inputs can be driven by the data bus, but the output only one of the output should drive the data bus, so there has to be a Tri-state gate, Tri-state gate between the output and the data bus, and the controller make sure that only one of the registers or one of the load that drives the data bus, ok.

And another important point is that that you see there are three control signal at the registers, all registers and one is clock which goes to all registers and there is a register A enable for register A means that if the output of the register A has to come to the data bus then the controller will give that enable signal high and data is enabled on the data bus. Similarly if there is a data on the data bus which need to be latched onto the register.

Then the controller give this register A latch as a level signal and when the clock head, the positive clock head come the data get large on to the registry ok. So that is our operation, that is true for this temporary register, you see there is a large signal and there is a clock signal. When the latch is 1 and the clock is you know active clock edge the data get large. This very

convenient way of latching the data because you could think about the way you could think of the controller clocking the register maybe say controller makes any this clock is 0 for a register and controller make it 1, then make it 0, then it gets a kind of a positive edge and data get latched ok.

But there is an issue there because you have to load something to a register every clock cycle then you have to know calculate 1, then 0, then 1 and so on, but in this case it is enough to keep it high for integral number of clock period and either high for 4 active clock cycle than 4 times the continuously data will loaded ok. So this came as such an advantage, so this shows the instruction register.

Because we said that the execution take multiple clock cycle. So this wholes the register value there. So that because is not a single cycle, so that they just hold the instruction that is decoded and it goes to the controller ok. Now when you look at the program counter you see the program counter is output is the one which is driving the address first and sometimes the stack pointer drive the address first.

You know that when an instruction like a call comes the content of the program counter is pushed on to the stack and stack means the part of the memory at that time the address of the memory is given by the stack pointer ok. So the stack pointer drive the address bus and the address value on the program counter is driven to the data bus and we goes to that memory location which is addressed by the stack pointer ok.

So that is this program counter and stack pointer, but you know that the program counter is incremented after every clock cycle, but 1 is here is that the program counter is 16 bit but the database is only 8 bit. So if you have a jump address kind of instruction code of gems and that is larger than 1 bite of the address come then second bite of the address come. So when 1 bites the comes there has to be kind of temporary stored.

And then second bite is along with the stored bite is loaded onto the program counter. So the loading has to happen byte by byte similarly when the program counter containers has to be pushed on the stack, the program router wireless split into 2 part, and has to be pushed bite by bite, so it involves lot of kind of multiplexers to select various paths. These are the control signal ok.

Now all these control signals are given by this control ok. So the controller is the 1 which gives say the large signal to the various register it is a 1 which gives enable signal for the output like register A enable, ALE enable, disk controller is a one which gives the various combinational selection in would like in this case of ALU whether it is in addition operation or subtraction or logical or comparator.

All that is specified by this the controller. So in a digital system design we divide this in 2 part. One is the data path where the data moves ok. So here is a data moving into the distance from register with the temporary registers and from the 2 ALU back to register and sometime the data come to the program counter or stack pointer, sometime data moves from the program counter to outside the memory and so on.

So all these part were the data the computation happens that is called data path, ok and this is the control of which controls the data path which give the latch signal, which enables various paths to markers, all that is called the controller ok. And the controller does not do any complication, controller just sequence or operations correctly because the complex there are lot of blocks and everything has to be done in an order in a sequence.

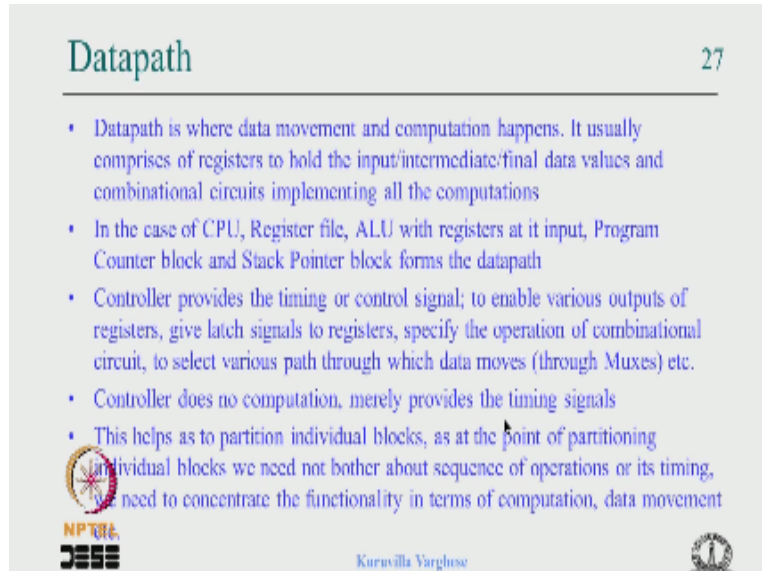
That sequencing is done by the control of through control signal, it does not computation, all the competition is done by the data path ok, that you have to remember. Now the question is that this kind of scheme allow you to kind of partition the design in a in a functional boundary without worrying about the sequence of operation like when you divide these the whole design into blocks.

You do not need to worry that for the for the what happens when the add instruction comes what happens when the comparator comes which way the data moves all that can be controlled by this control signal ok, so this kind of separation of the controller and data path allows us to partition the design properly ok. So that is a level 1 design, so we have to worry about the function at this point of the various block, the timing of the signal between them, like what is the timing of the control signal at the electrical spec.

And what is a source current, what is the what are the voltages and so on all that need the address at this level. Now like at this point we are not in a position to say the design is

complete because ALU complex block even registers are not broken down into kind of non pieces or higher level blocks as we call, so this has to be further designed ok. So let us look at the register ok.

(Refer Slide Time: 37:09)



Datapath

27

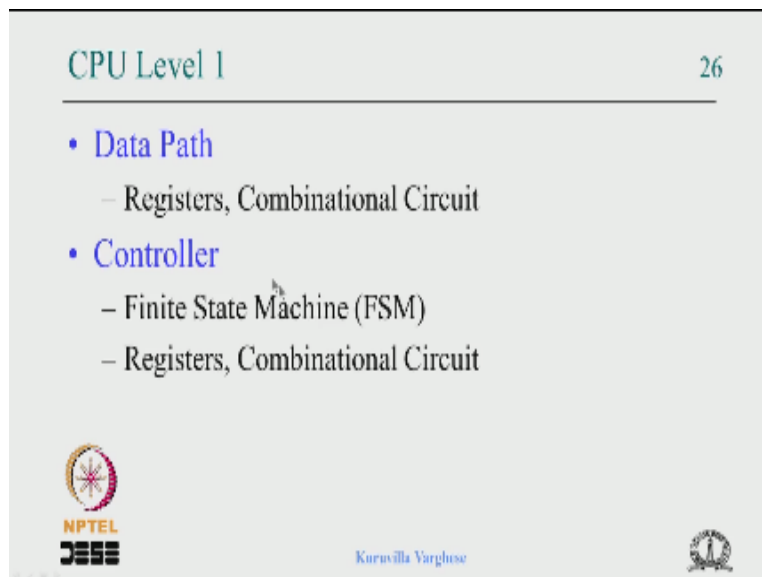
- Datapath is where data movement and computation happens. It usually comprises of registers to hold the input/intermediate/final data values and combinational circuits implementing all the computations
- In the case of CPU, Register file, ALU with registers at its input, Program Counter block and Stack Pointer block forms the datapath
- Controller provides the timing or control signal; to enable various outputs of registers, give latch signals to registers, specify the operation of combinational circuit, to select various path through which data moves (through Muxes) etc.
- Controller does no computation, merely provides the timing signals
- This helps as to partition individual blocks, as at the point of partitioning individual blocks we need not bother about sequence of operations or its timing, we need to concentrate the functionality in terms of computation, data movement

NPTEL
JEE

Kuruvilla Varghese

So when you talk about the data path.

(Refer Slide Time: 37:15)



CPU Level 1

26

- Data Path
 - Registers, Combinational Circuit
- Controller
 - Finite State Machine (FSM)
 - Registers, Combinational Circuit

NPTEL
JEE

Kuruvilla Varghese

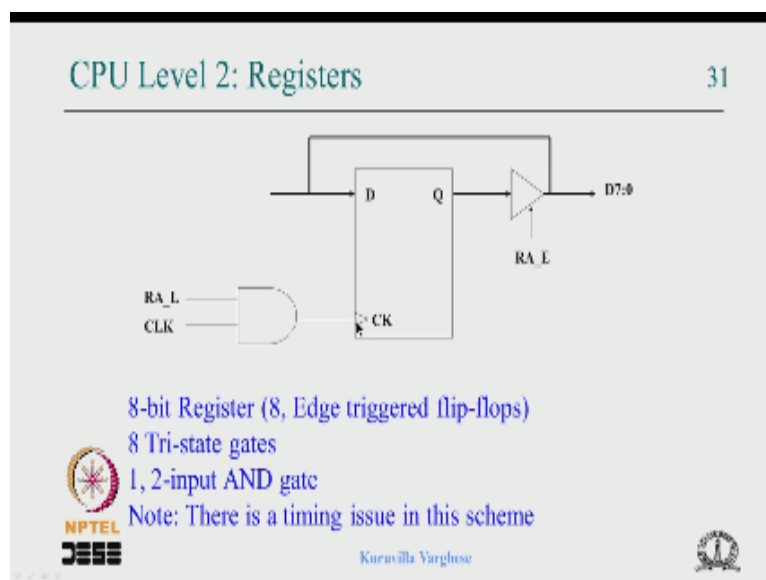
I want to mention that data path compose of registers and combination circuit in our case there are register, the ALU, there is combinational circuit within the program counter controller is something call of finite state machine and that also is composed of registers and combinational circuit. We will see that ok and one may be one of the important issue is that how many controllers are required for a data path normally ok.

In a simple case we can say there is only one controller required, but in a data path there are 2 asynchronous activities going on that means there is some operation in a part of the system which is not happening in synchronous with other part then you need a separate controller for that 2 mutually asynchronous part ok, but in principle if you have everything is kind of synchronous with 1 clock you need only one controller.

But then the controller operation can be very complex, so but when there is complex it is involved we might divide the data path into multiple blocks and you could have multiple controller for each part and you can our top level controller controlling this the level 1 controllers separately. So you can think of a hierarchy of state machine in the case of complex design ok.

So let us look at these registers, how to design them and when you take a register you should know that this is a 8 bit register, it involves 8 flip flops and the easiest thing to handle is output enable, so since only one load can drive the data bus, we can imagine this to be putting 8 tri-state gate at the output of the registers and control by a common enable signal which comes in the state machine.

(Refer Slide Time: 39:56)



But what about the input and only the latch signal comes okay ok, so let us look at the symbol design for a particular this particular register as a level to design ok now we're looking at this one of the day ok now we are looking at this one of the registers, so this would not possible theme of designing a register so you have it flip flop and Q7 to Q0 is going to 8 Tri-state gate

the enable of 8 Tri-state gate is controlled by the common register a neighbour and you see and just do not be confused with this kind of connection.

This just shows that input and output is connected to the same data bus that is all, my mind is do not think that is some kind of feedback or something like that it shows that the data bus same and now what we want is at latch signal along with the clock or enable the latching of the data the input ok. So when the clock edge comes and the latch signal is high then the data gets latched.

So we can say this and that together latch is clock is qualified by the large ok, it is a simple scheme in this awesome timing issue we will discuss that later, but if you look at it the design is over because we have broken down the register into 8 flip flops, 8 Tri-state gate and one AND gate. All these are blocks are known blocks at the high level even very simple circuit.

So at this point the design get over, there is nothing further design. So at the level 2 the registered design get over, it is in 8 bit register, 8 Tri-state gate and 1 input AND gate ok. So now let us go back and look at this program counter ok, how to design this program counter. So as I said that the program counter the program counter 16 bit program counter ok. So you can imagine already there is a 16 bit register which is holding the program counter value.

And that is driven that is present and this Q2 to 1 mux control when the program counter need to drive it is selected this path when the stack pointer need to drive the controller will give select this path ok, but internal if you look at the program counter operation when I say anything like jump and address come say the jump get large a register the address come by quite know the least significant bit comes first.

As I said that has be stored in a temporary register then it cannot be loaded the into the loaded into the part of the program count because then the address is kind of wrong address because one part whole old address and one part was a new address that cannot be done. So it is stored in memory register and when the new the most significant bite comes I think that along with this stored value can be loaded parallel to the 16 bit register ok.

So you can see that there are in would shall I park which selects various input to the program counter because you know that the power on reset the program counter as to load with some

specific starting location okay. So that as for the input of the program counter register will be different part, 1 part could be from the data bus, 1 part could be from this reset value. Similarly when interrupt comes the program counter is going to some address location.

That is a input program counter register. Another operation happens in the program counter is that after the execution of a instruction the program has been implemented ok. So that implement there is a increment insights after every instruction. The program counter is increment and loaded back okay that increment value loaded onto the program counter register.

So and ultimately when it drives the program counter output drives a data bus it has to be true at right side bus, so this enable ok and when in an instruction like call the current at the program count of values stored in just back in that case there is a 16 bit value we just go to an 8 bit bus you know in 2 steps and that is kind of control by this output select. So my point of discussion as that if you know of the program counter.

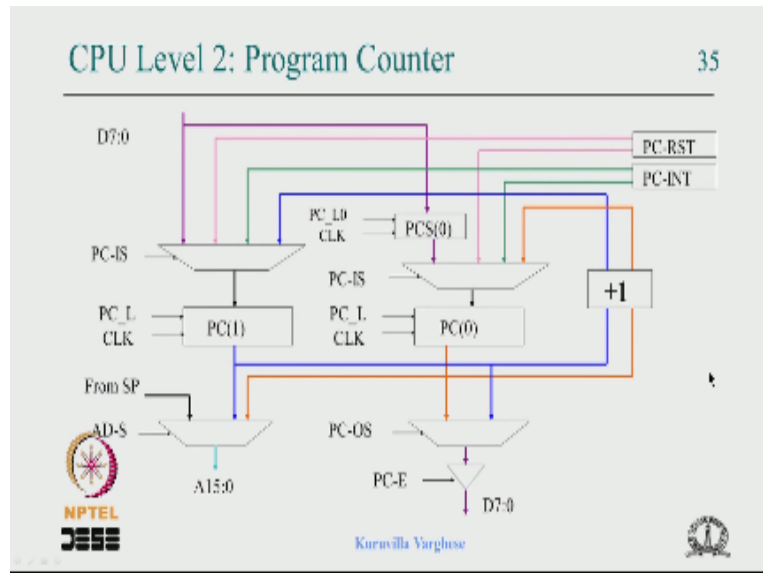
And if you describe it you can already know what is what is the internal structure of the program counter, so knowing the operation of the program counter and knowing the various then we can in for the blocks how it is interconnected and we do not need to worry too much about the timing because the timing is given by the controller that can be worried up and that can be handle much later.

So when you design a block at the level 2 you need to know what all the block does and you should be able to design that blocked. So let us look at the program counter design ok now maybe I will skip this particular design for a register ok. So I have shown a possible implementation of register there is another possible implementation here which is like you see the earlier cause the latch signal was kind of handed with the clock.

But in this case the latch signal goes to a 2 to 1 mux ok and when the latch signal is high the input comes to the to the flip flops are registers and the clock when the clock comes in gets latched, but you see the clock is not control clock is continuously clocking the flip flop. So at every clock the input the latched, so when the latch is 0 we have to give the input to the register as what is its output.

Otherwise to get corrupt so we feedback or re-circulate the output back to the input when the large signal is inactive. So this is a very nice way of controlling the register very useful way or the earliest scheme has its own problems as I said we will discuss it.

(Refer Slide Time: 47:01)



So let us get back to the program counter and we have described the operation and let us look at the program counter so when it comes to the level to design of the program counter here looks like this and I would like him that the program counter dresses 16 bit register and that is broken into two part the most significant byte call PC1, the least significant byte call PC0, both get clock.

And latch signal so that the input can be latched, ok now you can see this the most significant part is blue line 8 bits and least significant bit is red line which is a 8 bit. Now you can see that this 8 bit and this 8 bit goes to a 16 bit increment or a plus one circuit and it goes back to the respective program counter sections ok. So normally after running in section is executed this particular path selected.

So that program increment value of the program counter is loaded back to the program counter ok. Now this part this green path shows how the program counter is loaded with a new address like in the case of a jump or a call in that case the least significant byte of the address comes here and that comes to a temporary register and that has a latch signal and the clock signal coming from the controller.

So what the first byte the large signal is given and the up on the clock at the least significant bit get latched here and when the most significant byte comes the controller select this particular path green path by this input select line of the marks this path. So this most significant byte come here, this temporary register just byte comes here and the controller gives a latch signal and it gets latched ok.

So that is what is this path is about and similarly upon the reset the program counter is loaded with a specific starting location, so that red lines show that path. So at the time of reset the controller give select this red path and the recent value get loaded. Similarly when the interrupt comes the appropriate the timing appropriate time this reset interrupt location is loaded into the program counter by giving the proper input select, by giving the latch signal and so on ok.

So that is about the input that is all this 4 to 1 mux use one to choose the increment of part 1 choose the path from data bus, one to choose the reset address, one to interrupt address okay. Now we can see that the program counter drives address bus through 221 marks which is the 16 bit and the other port is coming from the stack pointer and the select line is coming from the controller ok, that is address bus.

And we have said that when there is a call instruction or interrupt instruction because it goes to that subroutine and it has to return back to the main program. So the return address has be stored somewhere. So in that case normally this goes to the to the memory ok. So then like we have to drive the values on program counter 1 and 0 back to the data bus and that just be done byte by byte.

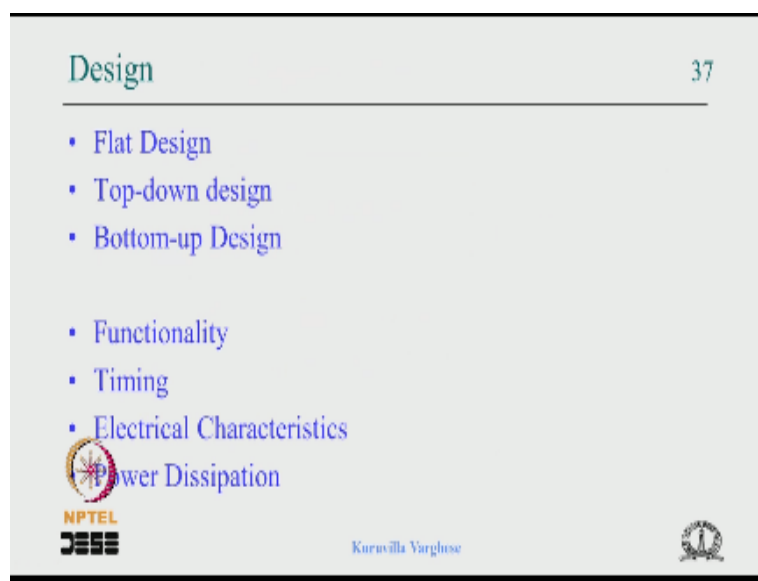
We can see that the orange line goes here, the blue line goes, so when such a scenario comes the controller gives an output select line to select the least significant byte and enable the tri-state gate and then it goes to the data bus. The next clock cycle this path is selected and in drives data bus. So that is the design of the program counter , as you knowing the operation of the program counter we are able to kind of break into pieces and we are able to design the data path through this means of the block.

So essentially it is composed of three 8 bit registers two 4 to 1 multiplexer which is each part is a bit. So we can say it is an 8 bit, 4 to 1 multiplexer two numbers ok. Data has a 16 bit

increment, it has a 2-1 16 bit multiplexer, it has a 2-1 8 multiplexer, it has 8 Tri-state gate. So at this point of time all the blocks and we know these are register, simple register, their marks increment Tri-state gate.

We can say the design is completed, we might know the code the design in a hardware description language or we draw schematic does not matter, we are broken down that complex block into simple blocks which is known high level combination blocks like the mux coder, encoder, decoder, adder, subtractor. Then you can be we know the circuit than that can be design.

(Refer Slide Time: 52:36)



So that is how we design from top to bottom maybe you can handle the stack point of yourself you can give a try with you not try at that and so have looked at the flat design where there is a symbol counters design, a top down design in the keys of the CPU and that is as I said any complex systems design top down, there something called a bottom of design which is like say you start with a very simple block like say you design the program counter then you come you design the register.

The new design try to integrated which does not work in practically but what can be done that you have tried to design the CPU you are made the level 1 diagram then the level 2 diagram then at that point you are not very clear about the program counter maybe you can start bottom up by putting the blocks. Otherwise all complex design has to be come down otherwise putting worth.

And then at each level up to worry about the functionality, timing, electrical characteristics, power dissipation and so on ok. So that is in a nutshell how you design a complex circuit. So in this lecture, today's lecture we have looked at I have jump from a very simple design to very simple complex design we have looked at the design of CPU and we said it cannot be handled in one shot like a flat design.

It has to be partitioned into non functional blocks and then each block has to be further maybe partition and designed to a level where we know the blocks you not to the level of registers and non completion block, then we can stop this design ok. So that is top down design, that is applicable to any complex system and we have looked at the level 1 diagram of the CPU.

And we have looked at what is the data path, data path is where data operation happens I know it is composed of registers and combinational circuit in the CPU you have most the data moving between the memory and the instruction register or between the various register ALU and back to the resistance between registers and memory and so on and the memory and the program counter all that is why the data movement happens.

So all that forms data path and then there is a controller with controls the latching of the data, enabling of the data to the data bus choosing various path in the data path. All that is the job of the controller and it does no computation and controller does absolutely no competition, it coordinator sequence operation in the data path, so this kind of divisional allows us to concentrate on functional partition without worrying about sequencing right.

We can handle the old sequencing timing of the control signal later when the controller is designed, knowing the operation we can design the block into pieces and then we can handle the timing. Then after that we have looked at the registers how to design the register we have broken down and bring it into a kind of some gates and the flip flops then we have looked at the various operation of the program counter.

And we come out with the scheme with the program counter is designed in detail and we have you found that it is composed of registers and multiplexers and Tri-state gates various different type of multiplexer and that what we are going to see in the design we will see lot of

multiplexer which is choosing various parts ok. So that is off you kind of partition and go from top level 2 top to bottom and design the circuit.

And in the next lecture now we look at the important part of the controller, what is the behaviour of the controller and what is the structure of the controller, how to find what is basic idea by which that control structure works and how to design, how to design a controller and all that we will study in the next lecture. So please go back now we are starting the design seriously. So you may not be used to designing complex designs.

So please revise and if you are not familiar with the microprocessor read some book on microprocessor and grasp the fundamentals of the microprocessor, so that you can follow it very nicely. So please work on this topic and I wish you all the best and thank you.