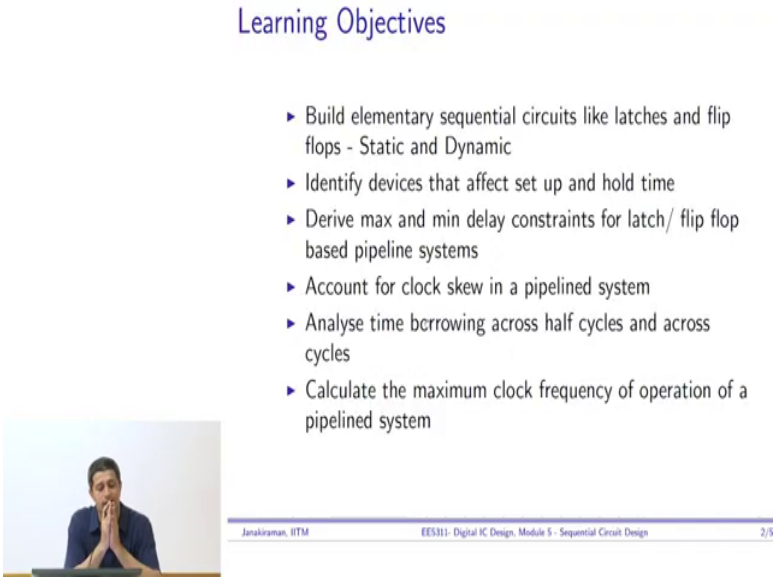
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 66

Introduction to Pipelining

As usual let us start with the learning objectives of this module ok.

(Refer Slide Time: 00:21)

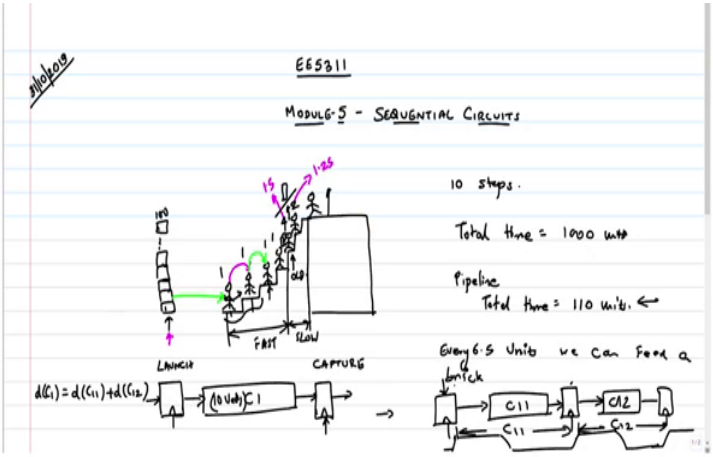


By the end of this module, you should be able to build elementary sequential circuits like latches, flip flops, both static and dynamic. So, just like we had static CMOS logic and dynamic logic; you have static flops, static sequential elements and dynamic sequential elements right, then you should be able to identify devices that affect setup and hold time ok.

First, you should be able to define what setup and hold time is that we will do; then you should be able to identify devices so that you can either reduce the set up time or reduce the hold time or whatever right. Then you should be able to derive the max and min delay constraints for a latch and flip flop this pipeline system ok; pipelining is probably one of the biggest takeaways from this module.

So, then accounting for clock skew in a pipeline system and then you know analyze; analyzing time borrowing. So, this is a very interesting concept that you can exploit to actually gain some time ok, but you should do it only with a; you can do it only with a latch based system, you cannot do it with a flop based system ok. So, we will look at that and then finally, you should be able to calculate the maximum clock frequency of operation of a pipeline system.

(Refer Slide Time: 01:39)



So, this concept of pipelining is really not very new to VLSI or it was not invented in VLSI at all right; this comes from a manufacturing line where you are doing things one after the other right. And the idea is you do not have to wait for the entire manufacturing line to be done before you start manufacturing the next one right.

But there is a more much more simpler example that all of us would have seen in our own daily life right. So, let we start with that ok; there is a construction site ok, they are trying to build the building has been built up to here and they are trying to build something from here on right. So, there are a lot of bricks which are kept here right; let us say there are about hundreds of bricks that are kept here and these bricks need to be transported to the top so that they can continue their construction from there right.

So, what do; what do these construction workers typically do? They basically stand on the stairs right. So, you will find one man standing or one woman standing like this everywhere on each stair right and finally, you have one guy on top right. The aim is to transport all these 100 bricks 1 to 100 up to the first floor without dropping even one of them; if you drop it then it's gone right; that is the aim. So, what do people do? The safest way to do this is to simply give one brick right and let us assume that it takes 1 unit of time for one guy to give it to the next and all times are equal for now.

So, if I give one brick; then it is going to take n time units for the brick to go all the way up there assuming there are n steps right and the next brick will be given after that this is the safest way right. But clearly after the this guy has handed over a brick to him or her then this guy is idle right. So, it really does not make sense in order to wait till the brick reaches on top. So, what do you do? As soon as this guy is done; you send the next brick from here to this person right. So, then by then this guy will give it here right.

So, effectively what you are doing is every; so let us calculate the time if we read it in that naive way; how much time would it take suppose there are 10 steps. Let us say there are 10 steps, you give one brick wait for it to go all the way; give the next brick. So, each brick is going to take 10 units of time and total time is 1000 units right. Now, on the other hand; if I

use this you know exploit the fact that there guy is idle and I just start feeding a brick; as soon as that guy is done, how long would that take? 100? Exactly. So, if I pipeline that this concept is called pipelining then the total time is equal to 110 units ok.

Now, this is not exactly representative of what we see in VLSI or in circuit design. The reason is we have made a very important assumption that all the delays of every person is the same right. In VLSI typically it is going to happen that this guy is probably very fast, this guy may be slow; slow, fast.

So, there is a; so what happens is you hand over a brick, he turns around and holds the brick until the next guy can come and take it right. So, that is where the delay mismatch happens; humans are very adept at adjusting that speed automatically right, there is a feedback which you will adjust it so that; that person comes and takes it correctly.

Logic gates do not do that; they just work at a fixed delay and the output will be held until the input changes right. So, you should ensure that this guy can turn round and hold the break and you do not give another brick by then for him to take it. Because this is now an automated system at some point if there is a mismatch, then you will drop the brick right.

So, let us assume that there is now one very old man who is very slow here right. So, what do you do you each guy now has to adjust his speed so that they work at the speed of the slowest guy. So, what will they do? Each one will turn around hold the brake and they have to hold it for a very long time right; so this is a problem. So, let us assume that these guys happen to be young guys or whatever who were quite fast right; they can turn around here to brick quickly and then the old man is standing here.

So, what do you do? Instead of asking this guys to just hold this brick all the while there; you say I will put a pedestal here you keep the brick here; then the old man can collect the brick from there I do not have to hold it for him there right. Similarly, if there is a you know another two people, I will put another pedestal here right. This pedestal is nothing, but a sequential element because it is able to hold the data for you until that guy takes it; until the

next one arrives right, it is not until the next guy takes it until the next one arrives right; the assumption here is that pedestal can put exactly one way.

So, the second brick comes the first brick is gone and then you drop it and you have you are failed right. So, clearly is now I introduced an element like this a sequential element or a pedestal in between I am going to increase my area because I need to put a pedestal there. I need space for that real estate for that; then there is also a delay which is involved in placing it on that pedestal careful right.

So, you go keep it and it is not even the show it and then leave it right you got to come place it. So, there is an overhead involved in; there is a time overhead involved in placing the pedestal like this or a sequential element, that time element right overhead is basically what we known as set up and hold time right; set up set up time is before the clock edge arrives, your data should be ready right; that is to say if I am going to place it on a pedestal before that old man can turn around and take it; then brick should have already been placed there it is not like; as he is turning around you come and place the brick there; then there is a chance again it will drop on the way right.

To ensure perfect functionality, the brick should have been placed a little ahead of time right and the whole time is a little bit to difficult to visualize in a physical scenario like this. So, we will come to that until later right, but what I am trying to say is I have a pedestal here now right. So, now how often can I send the next brick thing?

So, this scenario is too good basically where every gate has the same delay one that is impossible to achieve right; so forget about that. So, I have to place pedestals I have to put I have some overhead time and all that stuff. Now, how often can I feed the next brick him.

So, let us say hypothetically that I will remove this pedestal, this pedestal also right; hypothetically my old man is here, these guys are very fast, this last stage is going to be slow right. So, I am going to put a pedestal here; now tell me how long will it be before I can send the next brick into this system here in the entrance.

Student: (Refer Time: 11:20).

Yeah? So, let us say each of these guys take 1 unit of time to do this 4 guys right and these guys let us say he take 4 units and 2 units right. So, when can I send the next brick in is what I am asking.

Student: (Refer Time: 11:47).

Yeah? 3?

Student: (Refer Time: 11:55).

So, this guys will take 4 units of time to go and place the brick there, but how long will it be before that brick is taken off? No, no; so now, remember that I am not going to rely on this perfect communication of things; you know this guy giving it. So, I need a pedestal in order to ensure perfect functionality; I cannot rely on guys handing over a thing like that and I cannot feed it in before that guy is done.

In the sense, if this guy takes 4 units to hand over the brick; I cannot feed it after 4 units, I have to wait till this entire thing is done so that now there is the next pedestal there I go keep the brick there of course, this itself is a pedestal here right.

So, I am going to take 6 units of time to transport a brick from this pedestal to that pedestal right. These guys will take 4 units of time to transfer from one pedestal to another. The next brick can be fed when? Can you feed it after 4 units? What will happen? You keep one brick at time equal to 3 equal to 4 right; that guy takes 6 units to take that and finish it right, by then 4 units has come.

So, clearly at some point because this delay of 6 units is larger than 4; you are going to have more than 1 brick on that in between pedestal right; it will take 4, 8, 12; so somewhere 6 and then at 12 maybe you are going to have a problem right or some later time right. So, therefore

I have to wait for the slowest path in my system before I can feed the next data or the next brick into my pipeline system.

The slowest path is basically this 4 plus 2; 6 units of time of course, now because I have put pedestals in there; I am going to have a further loss in time, you know the time that I can send it. So, let us say there is a half unit delay because of placing a pedestal or a sequential element. So, therefore every 6.5 units right; we can feed a brick into the system without anything falling or breaking and this way I can efficiently send instead of sending bricks every 10 units of time right which I remember I told you was the safest way to do it right.

Each guy is not worrying about something coming and overwriting the existing state right; I can now send one every 6.5 units of time good. So, now why not I put a pedestal here or why not I put a ok; this may not make it why not I put the pedestal here one more pedestal. Now, how long do I have to wait? 4 units plus something right, why not I just keep adding pedestal?

Student: (Refer Time: 15:32).

Huh? Forget area, let us assume you have king of area; can I keep adding pedestals and improving my thing to the extent where I can bring it down to very very small delay. Now, of course, this is 4 and therefore it; so let us say this is just 1.5 right, 1.5; if this is this 4 were 1.5 and this your 1.25; then can I bring it down to almost 1.5 plus sequential overhead?

Student: (Refer Time: 16:12).

Why?

Student: (Refer Time: 16:16).

Exactly, your sequential overhead will start killing you after sometime. So, therefore you cannot infinitely keep pipelining; you have to place it very carefully in order to get maximum gain right. Translating this to a circuit this pedestal is basically called your capture clock ok.

So, if I want to draw; this I have a flip flop I am assuming all of you have done itself like it. So, you know what a I mean this big design do you know what a flip flop is right; we will come to the circuit details of this later. Then, I have a combinational block C 1 that is going to take 10 units of time. This guy is the; sorry this guy is called a launch flop, launch flop; this is the capture where you go and keep it eventually that is going to capture that data right.

And you have a combinational block that is going to have a delay of 10 units in between. If I want to pipeline this just like the way I did there; what I would do is I would now break this guy ok. It is like its guy now is a logic gate that has a delay and that is 10 units right of course, its partitioned as 1 plus 1 plus 1 plus 1 plus 4 plus 2 plus something maybe right. So, I am going to break this down into C 1 1, put another flop in between C 1 2, another flop in between that is my final capture flop. Of course, remember the same flop can be a launch flop for one circuit and a capture flop for another circuit right.

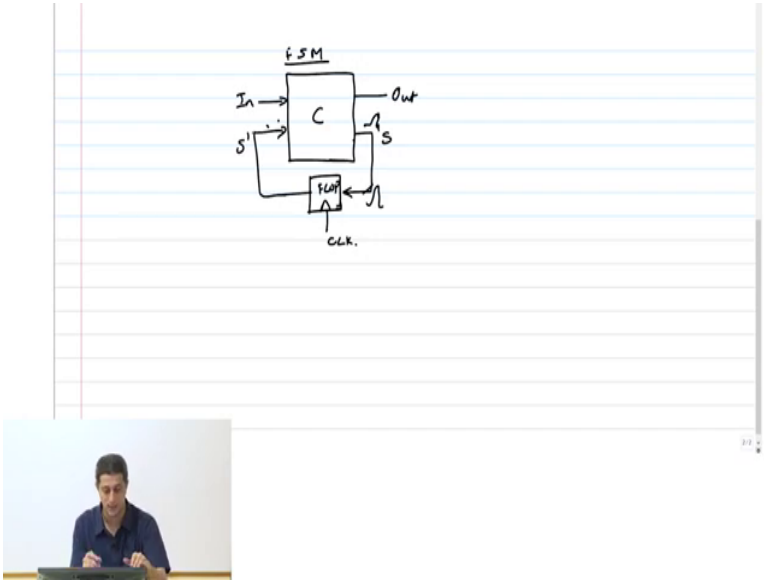
So, I cannot say that this is the launch and this is the capture because for the next stage that going to be the launch flop correct. So, the delay the idea is delay of C 1 is equal to delay of C 1 1 plus delay of C 1 2. And hence what is the maximum operating frequency of this pipeline system now or maximum time period? Just go by the analogy that we just get, in general I am asking. Yeah, it is basically determined by the whichever has higher delay that is going to get your clock frequency right.

So, therefore if I have a clock period like this; no, if I have a clock like this right; then what happens? My data should be ready before in each clock period; the data and the delay should be such that data is stable before the next clock edge arrives right. And therefore, it has to be simply the max of delay of C 1 1 and C 1 2 that will determine what this clock frequency is right.

So, this frequency has to accommodate C 1 1 right plus some overhead and C 1 2 as well. So, therefore if you accommodate it to the maximum delay, it will accommodate the minimum delay as well ok. So, before we go ahead and discuss more details and circuit implemented.

So, in this module we are basically going to look at the design of this block; we look at various circuit implementation of a flop, of a latch right and then a flop. Then we will also look at how to design systems with various timing constraints using these flops and latches. So, if I pipeline using a flop then what is my operating frequency, if I pipeline using a latch; what is my operating frequency right this kind of thing is what I want to do right.

(Refer Slide Time: 21:11)



And of course, all of us know what a Finite State Machine is FSM; you have a combinational block, I have some data here and this is some in, this is some out. And I want to feed data back from the output of the combinational block to the input of this forma national block right; obviously, I have to give it some delay right.

Now, question is can I just do put a logic gate here; let us say NAND gate, can I feed it back like this? So, what is the; what is the need for the delay first of all? Because any change here

will immediately affect this guy and that you come and override that state again right; there is an immediate feedback there. So, I need to delay that guy. So, if I delay it right; it means that this input will be held for some time that is what the delay means; the input will not change until that delay right, even if this input change is there.

So, if this input changes at some point, there is a delay through the gate after which this guy will change and therefore, that system can work. Question is can I just put logic gate like that and make it work? Thing is the delay that technically it is possible; point is that delay through these logic gates is now a fixed number. So, if I try to run it with a different; if I have to run it at the different frequency right, then I cannot do much the advantage would do using something like a flip flop is I can now get the delay to be actually my clock period. I can define how much delay I want right and make my system work at different clock frequencies.

Here, there is only one clock frequency at which the system can work; where you make sure that delay matches overall process (Refer Time: 23:26) and all that; then the system will see work reasonably, but there is only one delay and one clock frequency if you can work right. Of course, you have to give enough delay this is nothing; you cannot do like 30 picoseconds; an expect things to work right here to give enough delay so that you can do something with it ok.

So, in order to get a delay which we can control; we put the sequential element here in between and that is a flop or a latch and this will be controlled by a clock ok. This is a state S, this is the state S bar. So, this is all you design counters to do everything right. So, the need for sequential elements; we are actually seeing even a in a digital design course earlier ok. I just wanted to point out that this flop is nothing, but a delay element, but a programmable delay element; you can do how much or delay you want ok.