Embedded Systems Designs

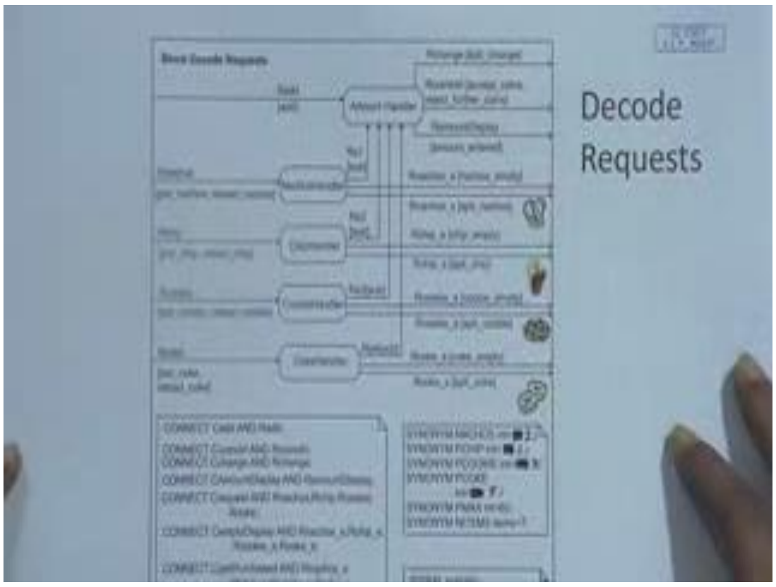
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 38
Data Flow Model – I

So, in the last class we were discussing about SDL representation of a vending machine, and we had shown there a number of functions; like the coin controllers, which accepts the coin decode request block, each of those who were blocks right. Decode request was a block that was decoding the request and was, looking at whether the correct amount of money has been paid, and there was another one which was a, coin exchange thing, and the decode request was also spitting out the actual product, when that thing has been, enough money has been paid for that.

Now, as we had said that SDL is the hierarchical model. So, we started with the top level where we had the blocks, and with each block we had the different channels right. We had shown the different channels through which the input output can come. So, the next level let us today first try to decompose that decode block.

(Refer Slide Time: 01:32)

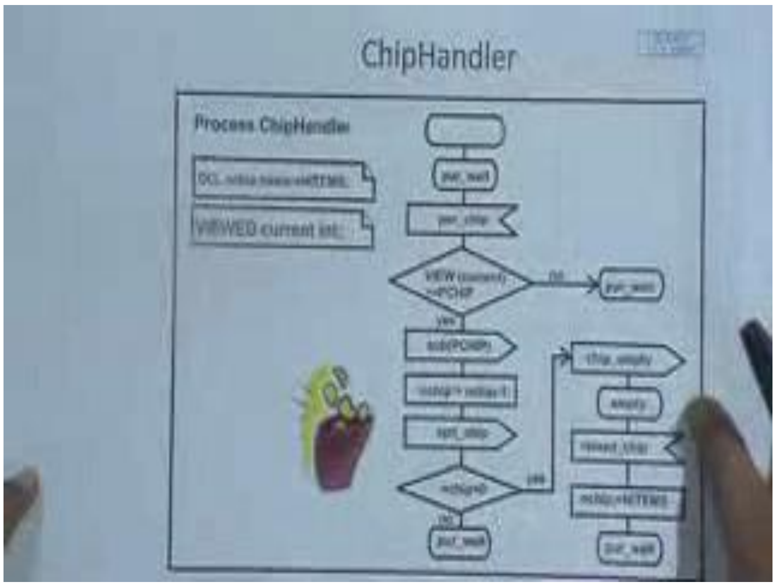


So, that decode request block is shown here. You can see, that this decode requests block, is being shown in terms of processes. So, these processes we have not shown in the earlier level of hierarchy right. So, here you can see that the block; that is decode request, is having a channel Radd which is receiving whatever is being added, and the variable that is being added is add, and that is the amount handler, and this octagonal shape means it is a process right. So, on amount handler what it happens, it rejects further coins, or it accepts coins, spits out change alright, and displays the amount. So, these are the, at the next level we are showing these details, that the amount handler what does it do, what are the channels it is connected to.

Now, you can see that in the earlier diagram, we had the channel which was connected at the top level, coming to this Radd, and this coin control. So, that is also by this connect, we are making an correspondence between the higher level naming nomenclature of the channel, and this lower level nomenclature of the channel. For example, when I am saying connect C amount display, and R amount display. R amount displays this one which is connected to the C amount display at that channel, which was at a higher level. So, in that way we have got the nachos handler, chip handler, cookie handler, and coke handler; the four things. So, this nachos handler is receiving the purchased nachos, that command that some button replaces, for selecting the nachos, or if the nachos are all sold out, then you will have to reload nachos. So, you reload nachos. So, these are two things that it takes, and accordingly it spits out nachos, or give the signal that the nachos are empty. In that way, we at this level lower level, we can show each of these processes, and here we are again SDL allows us to put in some variables also.

The variables is nachos, capital nachos means, it is an integer which is two alright; that means, the price of nachos is 2 rupees, price of potato chip is 2 rupees, price of cookie is 3 rupees, price of coke is 5 rupees. And maximum number of items is seven, for each of them or something like that, so we can put that right. We are also having a signal sub, whenever something is being given some one rupee has been paid, then this amount handler is subtracting that amount that has been given alright, and this one is for connecting now. Now, we are at the second level, what was there at the top levels. The top level was the block description, then at the second level we show the process description. What is the process? A process is nothing, but an FSM alright. Now let us look at the chip handler process and blow it down further.

(Refer Slide Time: 05:21)



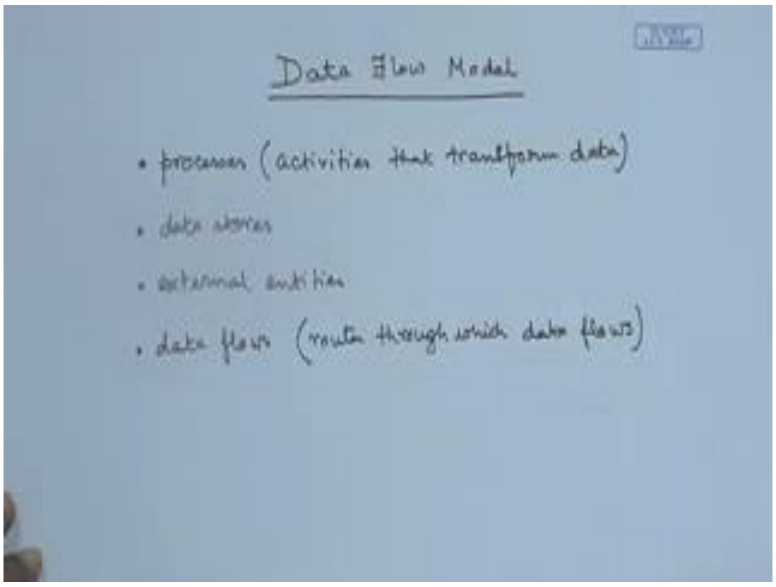
So, the chip handler will be something like this. There is a process chip handler, what is it doing, is waiting for the purchase. somebody says purchase chip, you want to purchase the chip, then sees whether the current number of chips are there, is whether its available, if not then you will have to wait, if yes then subtract one chip alright, then n chip minus 1. Now this part is coming when, when I am coming to chip handler. I am coming to chip handler when the amount. See I have just said purchase chip alright, but then you have to pay the amount; that is being handled by the amount handler. This one is the only handling the chip.

So, you see this purchase chip is an input and number of chips you are subtracting; that is an output, sub p chip, number of chips is being reduced, you are spitting out the chip; that is an output number of chip is zero, then you have to wait for the purchase. Purchase chips, yes. Now you see here this purchase weight means it is coming, now it is waiting for purchase chip, reload chip, is waiting for reload chip. Once the chip is reload, then the number will be shown up. So, here n chip items is in items, n chip; that is the declaration, that is equal to the n items, and current is an integer. What is the current, view current presently how many are there? So, these are different declarations. So, in that way the syntax you need not memorize, and there is no need of doing that whenever you need, but the main thing is, to understand the philosophy that, we can decompose it, at different levels of hierarchy, and this is basically an FSM, drawn in a different way, where it is, these are the outputs and there are the input.

Student: Sir, if there is no output (Refer Time: 07:37).

Then you do not need to give this. If there are no outputs, say for example, here there is no output, only input has come reload chip, you have put in an internal variable, and you are wait alright. Next so, with that, with this example of SDL, we conclude our discussions on state based representations. Next we move to another type of representation, which is known as the data flow model.

(Refer Slide Time: 08:16)

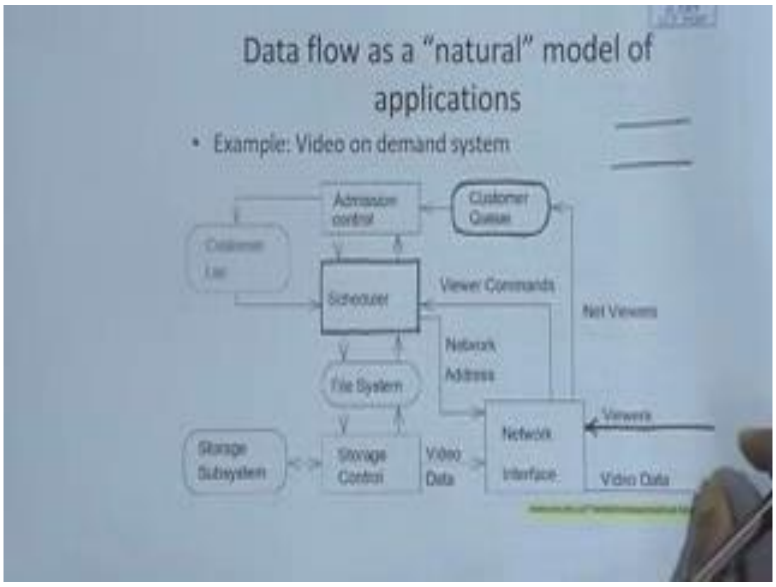


Data Flow Model

- processes (activities that transform data)
- data stores
- external entities
- data flow (route through which data flows)

Now data flow is very common, and very natural. In data flow model, we are not looking at the states, we are looking at how the data is flowing alright, across the system, and we will show a particular type of example.

(Refer Slide Time: 09:00)



Data flow as a "natural" model of applications

- Example: Video on demand system

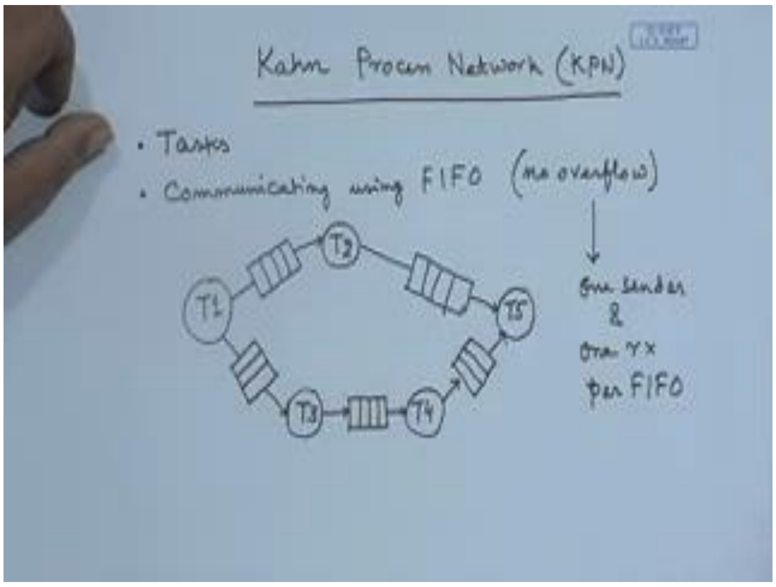
Let me see if I can show you; say one example of data flow, I will show you; that is a video on demand system. It is very common, you have all, in your software engineering classes and all those, you have seen the data flow graph, where you have got the data and the processes. Here you see, this is a video on demand system. So, what is happening here? The data is coming in, and these rectangles are, the rectangles are the processes. So, we, the viewers come to a network interface, and it goes to the customer queue.

Now, customer queue is a data holding capacity, sometimes in software engineering you show it like this right, where you put in some data right, buffer sort of thing. And from that customer queue it goes to the admission control. The admission control is a process, which is selecting the customers from the customer queue. All of them are requesting for video. It comes to the customer list, and then the customer. There is a scheduler process which is taking the customer list, and then it is giving the network address, it is coming to the file system. There is a storage subsystem, where all the videos are stored, then storage control, the video data comes here, the network address of the customer comes here, and the video data goes out right. So, here we are just showing the flow, we are not talking of the intermediate stage right. We are talking of the data flow, the data holders, and the processes that are dealing with data. So, this one tells us which particular process is dealing with which data.

Now, how do you define data, let us data flow. There are several definitions. So, I would not like to define alright, but a data flow model, consists of a couple of things, which are processes; that is, these are the activities that transformed data, from one form to the other. So, for example, here each of them, the scheduler is a process right, is a process. Then there are data stores, which store the data. So, the holding areas of data; for example, the example is again here the customer queue is, this one is a data store right. There are external entities; for example, here the external entities are the viewers alright, they are the external entities, and then there are data flows.

By data flows we mean the routes, through which data flows alright. That means, this entire path, a particular data, if I just talk about a particular data, I am concerned about, what is the path through which it is flowing, what are the processes to which it is coming as an input alright, or which process is generating as a output, all those things. We are a not talking about the states involved of course. There is a controller in between, but in this model we are not looking at. Next we look at a very popular nowadays, and it is becoming very useful also.

(Refer Slide Time: 13:57)


Kahn Process Network (KPN)

- Tasks
- Communicating using FIFO (no overflow)

A particular data flow network known as the Kahn process network, or KPN. A lot of research is going on now, modeling systems using KPNs. Now here each component of the Kahn process network is modeled as a task. So, we have got some tasks. Now this task can be programs or processes alright, and the tasks are communicating among themselves. And they are communicating using first in first out type of queues. And we are not considering any overflow; that means; essentially we are assuming that they are of infinite capacity. So, let us take an example of a Kahn process network. So, here is a task T 1. There is a typical task graph that we had seen earlier also, in another context maybe.

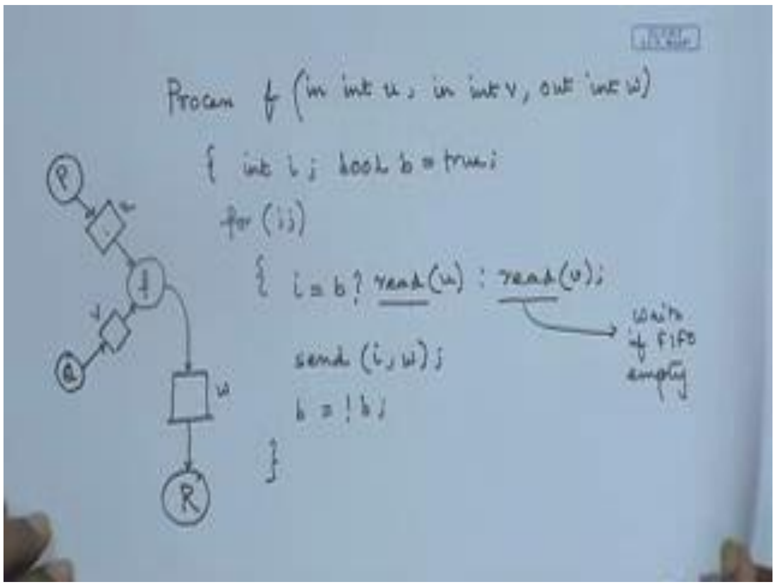
Now only one sender and one receiver are FIFO alright. Therefore, I will have one sender receiver here that is connecting to this. Similarly there will be one dedicated FIFO between these, right. I am not too much concerned about the size of the FIFO, while I am drawing it, because that is basically there will be no overflow, that is assumed in this model. So, for each of them, there is a FIFO connecting them, so one sender and one receiver, these are important, receiver per FIFO And since the FIFOs do not have any overlap, and because of this, one sender and one receiver.

If you recall the situation that we encountered with, in case of SDL, there was a conflict right, because every process was taking from one particular FIFO and a lot of processes could write into that FIFO. Therefore, it could be that process p one writes in the FIFO of p 3, and p 2 was also writing in the FIFO of p 3. Now the order there could be raised, and the order could have changed, and depending on which one p 3 is consuming, the result would be different. In this case we do not have that situation, because we have got individual dedicated FIFOs, alright. That is the basic of Kahn process network that we have. Now so, let us write down one example for this.

Student: (Refer Time: 18:28).

Between two processes, we have got one FIFO; that this one is writing into this. Now if this one wants to write into it, then there will be another FIFO.

(Refer Slide Time: 18:48)

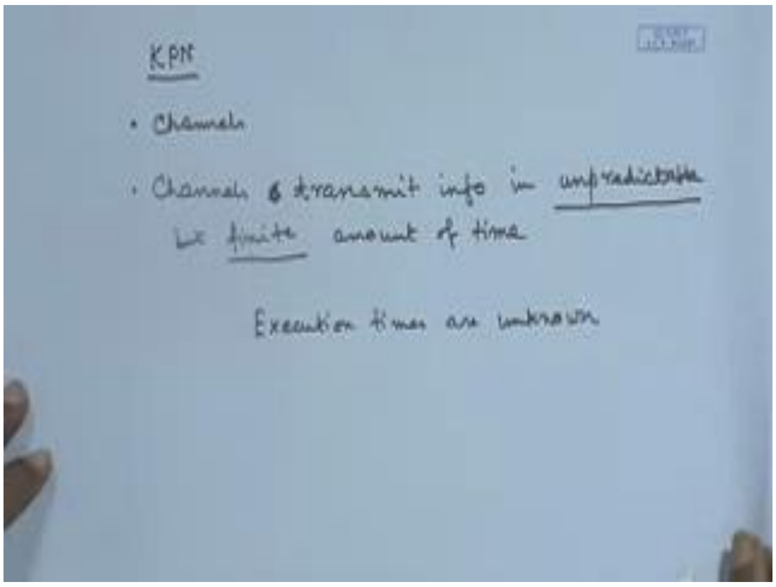


```
Process f (in int u, in int v, out int w)
{ int i; bool b = true;
 for (;;)
  { i = b ? read(u) : read(v);
    send (i, w);
    b = !b;
  }
```

So, if I write this, the process f which has got input is integer u, another input is integer v, and the output is integer w. Then maybe I have got two local variables int i and some Boolean b, which is signed to be true alright. Now I do in an infinite loop, I do, i will be depending on b, it will be either reading u, or will be reading v. So, how will that look like in the Kahn process. Now this read will return the next token in the FIFO; that is there, and waits if it is empty now this read, both these reads I mean, waits if the FIFO is empty alright, and then I have got send i to w. What is w? W is an output channel right, and then I make b assigned not, something like this if I do.

Now, what is this actually doing, what is this doing? This is, if I just draw it something like this, then this process, is have taking input, from either u or v alright. So, must be those are coming from two different processes. So, here my process f is here, and it is taking data, from maybe another process p, which is sending it through u, and maybe another process q, which is sending it to v right. And depending on the Boolean variable, it will either take from this or from this. And if that is empty, it is waiting. Even if it is there than the first one is taken, if the value is there, and then it is sending it to, maybe there is one output, there is one cube here, and that is w, this was u this was v and is going to some other process r, right. So, this is the data flow; that is captured in this. Now there are some. So, what is the big deal about it?

(Refer Slide Time: 22:29)



KPN

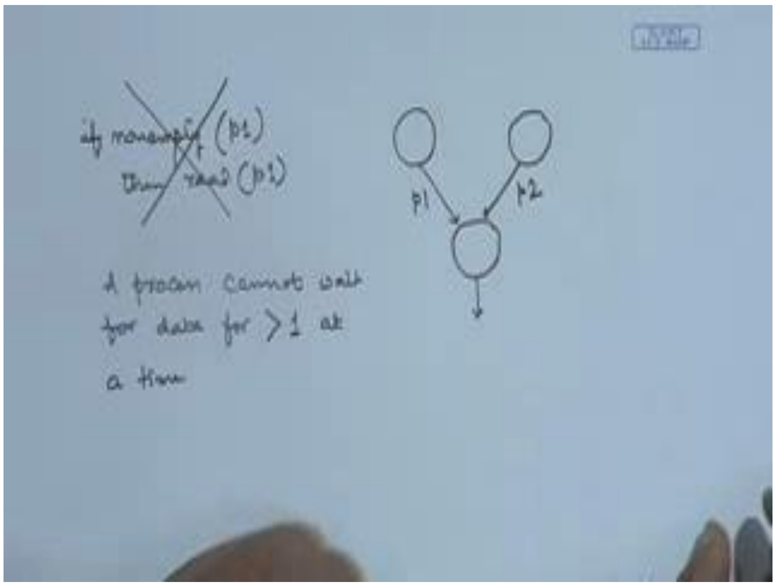
- Channels
- Channels transmit info in unpredictable but finite amount of time

Execution times are unknown

The big deal about it, is that first of all in KPN, communication is only via channels alright, no shared variables; that is number one. Mapping from one, more than one input to another, more than one output is possible. Channels transmit information, transmit information.

Now, this is important. In unpredictable we do not know beforehand. We do not know how much time it will take, but finite amount of time. So, that system will be live, because I am not taking into consideration anything about the implementation of the channel here. Therefore, in general, the execution times are unknown, in general. Unless I make a special case, where I say how much time it will take. This is one issue, that execution times are not known beforehand, the other thing. So, we are all doing it through channels. Now how does the channels communicate the data. So, let us look at another example diagram.

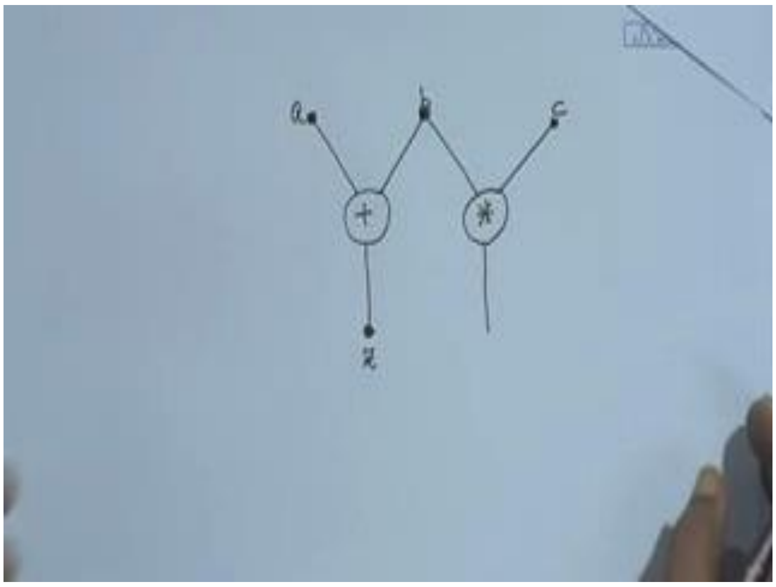
(Refer Slide Time: 24:25)


~~if nonempty (p1) then read (p1)~~

A process cannot wait for data for > 1 at a time

Let us show it through a diagram, say there is a task, there is another task, and they are communicating, and there is another task. Now there is one channel p 1, one channel p 2. A process, say for example, this process, cannot take for the availability of data before it commits a read. It is not like that I will perform the read only when the data is available. So, it will try to read, if there is no data, it means that read will fail. Basically the data flow model.

(Refer Slide Time: 25:17)



Let us diagnose a little bit, the data flow model that came, was something like this, that suppose there is an operator plus, and there is an operator multiplication alright. Now we have got data, coming from here, maybe this, maybe a b c alright. Now this is a typical diagram, that this operator is taking this data. Now, whenever a data comes here; that is (Refer Time: 25:54), I show it with a token, and whenever that data comes here. So, this operation will fire, whenever both these data are available, and maybe it will produce some data which may be z. And since b is available, this one is has fired, but might be c has not appeared, but this is ready, is reading and failing. So, whenever the data comes here, then also this one will fire; that is the data flow model.

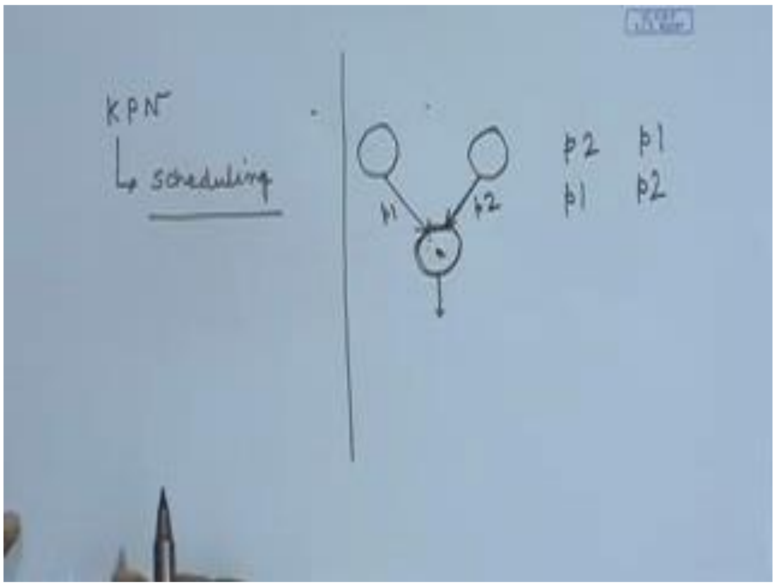
So, the control, is therefore, when the read will actually execute, when the read will succeed and progress, is entirely dependent on the availability of data; that is a little different from the typical control base systems. So, it is, I mean we cannot do something like this, if non-empty p 1, then read p 1 this sort of thing, we cannot. We are just doing this, and a process. Another important thing is a process.

Student : (Refer Time: 27:27).

Yes that is the data flow. See all these models; all these models are your representation of how you want to express your intention. So, this model is telling so much; say as if it is just like this modeling are just like expressing certain things in writing, what you want that you are showing through a model. Now please do not confuse it with the implementation. Now, this model is telling, giving you some information, it is has got a very clear semantics, based on that you can understand what the designer or the design system will understand what is asked for, and then how it will implement, whether you will put in some a handshaking signal or not; that is up to you. This does not mean that that there will be no controller in this system, but in this model I am expressing what I want. Now it is up to the synthesis system to decide on how this, they will be captured.

A process cannot wait for data, I mean for data, for more than one port at a time. I mean as soon as the data is available; that means what? That means, a process has to commit, to execute this data. A process cannot wait for data, for more than one port at a time. So, this process is waiting for this data, from this, as soon as that one comes, its fine ok.

(Refer Slide Time: 29:37)



Now, therefore, since we are doing this, in this sort of scenario. The order of read, does not depend on the arrival time, the order in which they will be reading, it will be purely on the data, alright. So, it is a data, if the data is there, whether I read this first or this first, it really does not take. Therefore, this Kahn process network is determinate, determinant e, because is not that order is not so much required in this case. So, that is very nice way of.

Student: (Refer Time: 30:26).

Which one, this process? No here this process is producing p 1. So, whenever it is producing p 1, this one is accepting p 1, is getting there.

Student: (Refer Time: 30:47).

No, that is this data flow means that. So, basically if this one comes it will exist.

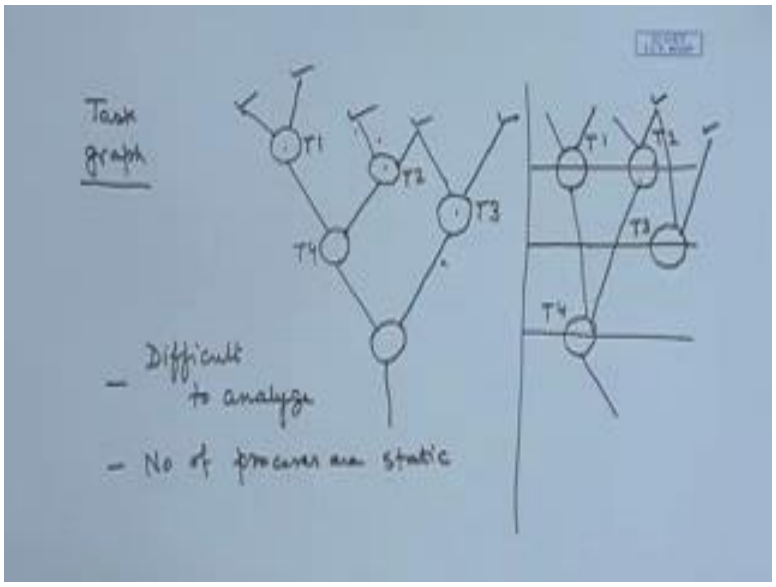
Student: (Refer Time: 31:02).

No we need both; therefore, p 1 has come. I am not executing, p 2 has come then I am executing. Therefore, it is not dependent on the order in which p 1 and p 2 comes.

Student: The one which we are waiting for only that will be executed (Refer Time: 31:21).

No that will be received. See p 1 has come. So, I will accept this p 1, I will accept the p 1, and then p 2 can come a little later. As p 2 comes I need both of them. So, if the order p 2 p 1 arrival, or p 1 p 2 is actually not meaning, not affecting; that is why it is determinate this, there will be no race in this case alright. So, this is fine, but the main challenges in this case, whenever we are trying to implement it. We have made a very tall assumption, that there will be no overflow over the queues that is very difficult to implement. Ideal it is not possible to implement; therefore, whenever we implement some KPN, then the challenge is to schedule that we have to. So, I will discuss later in detail, what is meant by scheduling of tasks, scheduling is very important. Scheduling means if I have got a number of tasks maybe. And I am just showing another data flow graph.

(Refer Slide Time: 32:36)



- Difficult to analyze
- No of processes are static

Am I showing it to the correct way? Suppose this a task graph, this a task graph. Now scheduling essentially means, at which step I will execute which task. So, maybe you can see that these; this and this, if the data is available. All these data are available, this one is available, this one is available, all these are available, then I can schedule all of them together alright, but due to that, may be the buffer, I mean the capacity may overflow. Therefore, scheduling means which of these I will schedule in which control step, one possible schedule could be, that in step 1.

Let us see this step one I schedule ask T 1 T 2 T 3. Suppose I task schedule T 1 here T 2 here. So, this one data is here, this one data is here, and T 3 maybe I schedule later, it is possible, T 3 I schedule at a later point of state. So, this one comes here, this one comes here. Although the data was available I am scheduling it later, because that is that is over and above the KPN, the KPN said that all these things can fire together and work together, but because of my implementation restrictions that I need to manage the buffer. So, I may like to schedule them separately and maybe this one which could be done; since both of them are ready it could be done at this step, I may like to schedule it later also, this T 4 can be scheduled later.

So, I will discuss in de detail what is meant by scheduling. Scheduling means, essentially I am just mentioning it here, that assigning control steps, assigning the execution steps to the tasks. So, for KPN that is a very important thing that it is a challenge, that we cannot accumulate tokens. So, therefore, we will have to proceed. The KPN's are very powerful, but one problem is, it is very difficult to analyze, it connects. What do you mean by powerful? By powerful I mean I can express, many of the tasks, many of the types of computation in this, but it is very difficult to analyze, because of the accumulation of tokens and buffering, alright.

Another point in KPN is, number of processes or tasks are static, so whenever the tasks arrive in, new tasks arrive in, I cannot handle that with KPN. So, with that we conclude the discussion of KPN and in the next lecture will move to another type of data flow which is synchronous data flow.