

**Computer Architecture and Organization**  
**Prof. Kamalika Datta**  
**Department of Computer Science and Engineering**  
**National Institute of Technology, Meghalaya**

**Lecture – 20**  
**Design of Control Unit (Part 4)**

Welcome to lecture 20; the design of control unit part 4. Till now we have seen the various internal bus architecture and we have also seen that how various instructions are executed. Now we will look into the approaches that are required for generation of these control signals; what kind of approaches are there. Broadly there are 2 types of approaches.

(Refer Slide Time: 01:00)

**Introduction**

- To execute an instruction, the processor must generate control signals for the data path in proper sequence.
  - Example: ADD R1, R2
    - a)  $R1_{out}, Y_{in}, SelectY$
    - b)  $R2_{out}, ADD, Z_{in}$
    - c)  $Z_{out}, R1_{in}$
- Two alternate approaches:
  1. Hardwired control unit design
  2. Microprogrammed control unit design

The slide footer contains logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

One is hardware control unit design, and microprogrammed control unit design. So, what we are trying to say, we know that for this instruction these are the control signals that are required to be generated.

(Refer Slide Time: 01:31)

© CET  
I.I.T. KGP

T1: PC<sub>OUT</sub>, MAR<sub>in</sub>, READ, SELECT<sub>4</sub>, ADD, Z<sub>in</sub>

T2: - - - -

T3: - - - -

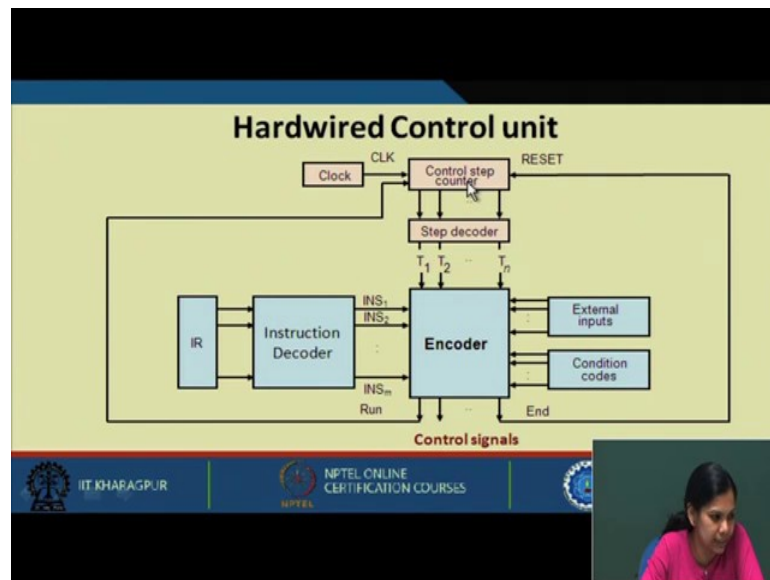
Control signal

T1:	1	0	0	1	0	1	1
T2:	1	1	1	0	0	0	0
T3:	1	0	1	1	1	1	1

But now these control signals must be generated in a proper sequence what do you mean by that. So, let us take an example.

So, we say that in T1; what is performed PC<sub>out</sub>, MAR<sub>in</sub>, READ, Select<sub>4</sub>, ADD, Z<sub>in</sub>. So, at T1 for all the instructions these signals must be generated. Similarly in step 2 some more signals, step 3 some more signals and so on. So, the processor must generate the control signals for the data path in a proper sequence. We will be looking into two approaches, one is hardware control unit design, and another is microprogrammed control unit design.

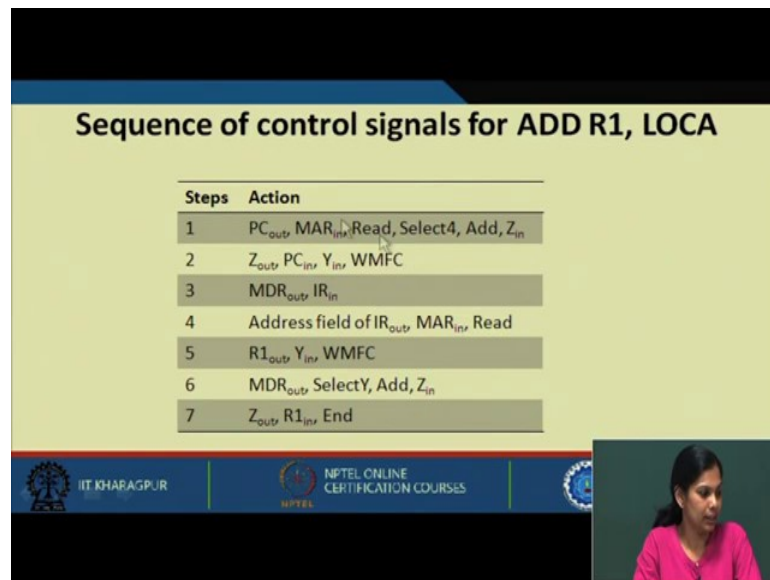
(Refer Slide Time: 03:16)



Coming to hardware control unit design, let us see what we have here. We have a clock that is hitting the control step counter. The control step counter is connected to a step decoder. Basically the step counter generates the steps that are required by an instruction to generate the control signals. So, at these steps T1, T2, T3 various control signals will get generated depending on which instruction we are using, what is the step of that instruction, whether some external inputs are required for it or not, if some conditional codes needs to be checked or not. So, depending on all these, we see that in this encoder the content of this step decoder the content of instruction decoder and the external inputs and the condition codes all are getting input, and the encoder is encoding based on that the control signals are generated.

So, we will be looking into each and every aspect of this instruction decoder. We already know conditional codes for every ALU operation; certain flags gets sets and depending on the condition codes can be considered external inputs what can be the external input an MFC memory function complete that is an external input that comes from a different module that is memory and there are 2 more signals one is Run another is End we will be seeing why we are requiring this, but you see this entire structure. So, this entire structure is basically doing what it is generating control signals. How it is generating control signals depending on which instruction and which step it is, and if there is some external inputs for that depending on all these this encoder will generate the control signals.

(Refer Slide Time: 05:51)



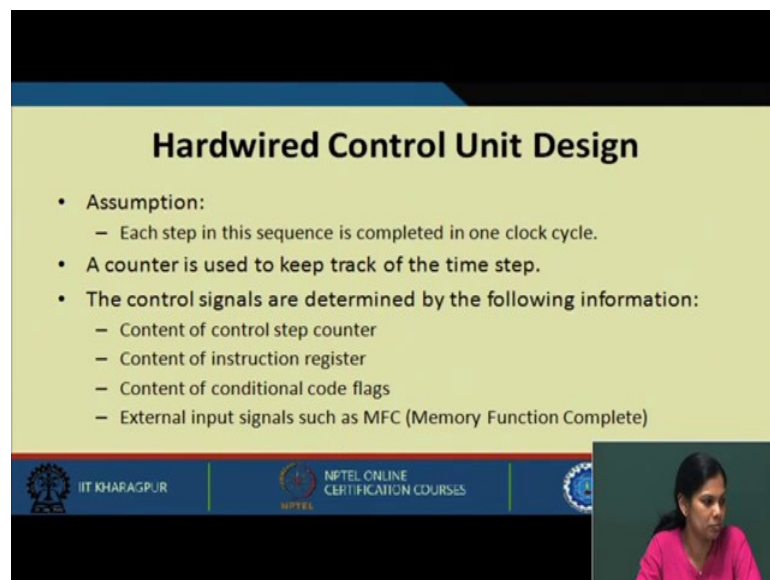
### Sequence of control signals for ADD R1, LOCA

Steps	Action
1	$PC_{out}$ , $MAR_{in}$ , Read, Select4, Add, $Z_{in}$
2	$Z_{out}$ , $PC_{in}$ , $Y_{in}$ , WMFC
3	$MDR_{out}$ , $IR_{in}$
4	Address field of $IR_{out}$ , $MAR_{in}$ , Read
5	$R1_{out}$ , $Y_{in}$ , WMFC
6	$MDR_{out}$ , SelectY, Add, $Z_{in}$
7	$Z_{out}$ , $R1_{in}$ , End

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, and this we already know that this is a sequence of control signals for ADD R1, LOCA. So, these are the set of microinstruction that are executed.

(Refer Slide Time: 06:08)



### Hardwired Control Unit Design

- Assumption:
  - Each step in this sequence is completed in one clock cycle.
- A counter is used to keep track of the time step.
- The control signals are determined by the following information:
  - Content of control step counter
  - Content of instruction register
  - Content of conditional code flags
  - External input signals such as MFC (Memory Function Complete)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

In hardwired control unit design there is an assumption that each step in the sequence is completed in one clock cycle, but remember one thing that when we are reading something from the memory it may not be completed in one clock cycle. We cannot ensure that if we are reading from memory it will be completed in 1 clock cycle, but this is an assumption that we have made. A counter is used to keep track of the time step.

A step counter is there which keeps track of the time step. Let us see the previous instruction, here 7 steps are required; some instruction might take 8 steps, some instruction might takes 4 steps or 5 steps. So, this has to be taken care of. So, counter is used to keep track of the time step. The control signals are determined by the following information: what are the following information the content of the control step, counter content of the instruction register, content of conditional code flags and external inputs such as MFC. We have already seen that the encoder is taking input of the step decoder, taking input of the instruction register, and it is also taking the conditional code and external inputs depending on all it is generating the control signals. So, that is what it is saying that the control signals are determined by the following information content of control step counter, content of instruction register, and content of conditional code flags and external input signals such as MFC.

(Refer Slide Time: 08:24)

- The encoder/decoder circuit is a combinational circuit which generates control signals depending on the inputs provided.
- The step decoder generates separate signal line for each step in the control sequence ( $T_1, T_2, T_3$ , etc.).
  - Depending on maximum steps required for an instruction, the step decoder is designed.
  - If a maximum of 10 steps are required, then a 4 x 16 step decoder is used.
- Among the total set of instructions, the instruction decoder is used to select one of them. (That particular line will be 1 and rest will be 0).
  - If a maximum of 100 instructions are present in the ISA then a 7 x 128 instruction decoder is used.

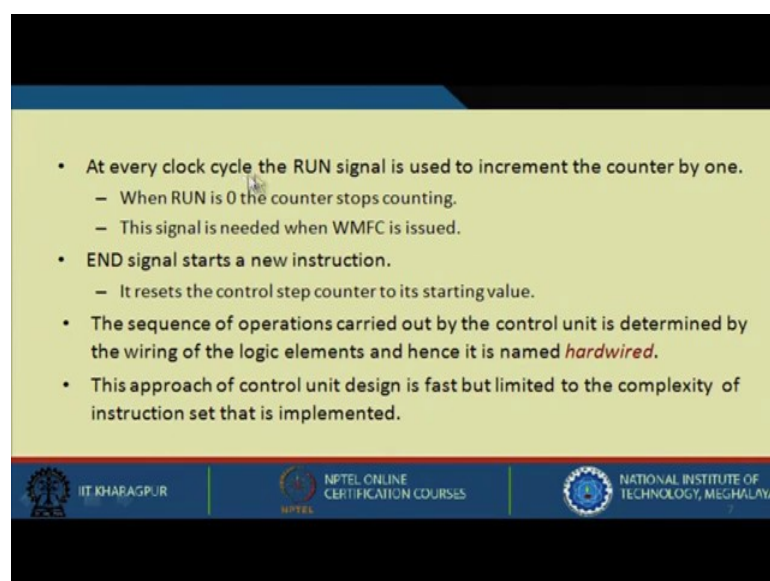
Now, let us see this; the encoder-decoder circuit is a combinational circuit that generates the control signals depending on the inputs provided. So, whatever be the input from the step decoder, instruction register, external inputs, and conditional code, the encoder-decoder circuitry will generate the control signals. Now see the step decoder generates separate signal line for each step in the control sequence. So, at step 1, PCout, MARin, READ, Select4, ADD, Zin happens. At T2, Zout, PCin, Yin is happening, and in T3 MDRout, IRin is happening. So, at various time steps different control signals are getting executed.

So, how many steps we require is dependent on the maximum steps required by an instruction. The step decoder is designed depending on the maximum steps required for an instruction. So, let us say we have 20 instruction in our instruction set architecture, and every instruction takes 4 steps, 6 steps, 7 steps, and maximum steps that is required for an instruction is let us say 8. None of the instructions take more than 8 steps, then what will be the size of your step decoder we need to generate a maximum of 8 steps.

So, a 3 x 8 decoder will do, but if the maximum step is 10 in that case step decoder size will be 4 x 16. In case we require only maximum of 8 steps for any instruction in that case a 3 x 8 decoder will do, but for other if it requires more then we will be requiring 4 x 16 decoder. So, if a maximum of 10 steps are required then a 4 x 16 step decoder is used. Among the total set of instructions, the instruction decoder is used to select one of them. That particular line will be 1 and the rest will be 0. So, depending on total number of instruction let us say we have a total of 30 instructions.

So, if you have a total of 30 instructions how many bits will be required to encode? It we will require 5 bits, because 5 x 32 decoder will be used. So, output at any point of time depending on the input any one line will be 1 and the rest will be 0. So, it can encode 30 instructions; similarly if you have 100 instructions how many bits will be required to encode that? We would require 7 bits and we require a 7 x 128 decoder.

(Refer Slide Time: 12:14)



- At every clock cycle the RUN signal is used to increment the counter by one.
  - When RUN is 0 the counter stops counting.
  - This signal is needed when WMFC is issued.
- END signal starts a new instruction.
  - It resets the control step counter to its starting value.
- The sequence of operations carried out by the control unit is determined by the wiring of the logic elements and hence it is named *hardwired*.
- This approach of control unit design is fast but limited to the complexity of instruction set that is implemented.

Logos at the bottom: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Then a 7 x 128 instruction decoder is used at every clock cycle. The RUN signal is used to increment the counter by one; when RUN is 0 the counter stops running this signal is needed when WMFC is issued. Now see what we are trying to say. So, when the step decoder value is at step T1 for instruction ADD, certain set of control signals will get generated. After that step is performed then we need to move to the next step that is T2. So, while moving to T2 what we need to do we need to move from T1 to T2.

So, the step counter will increment to T2 and now whatever is the assigned job for T2 it will get executed, and similarly the step counter will go on incrementing. So, this RUN signal is basically used to help the step counter to implement the steps of any instruction. Similarly when RUN is 0, the counter stops counting and when this RUN is required to be 0 this is required for WMFC; see when we are doing WMFC, we will not be executing the next one until we get this confirmation that the data is available, then only we can take that data because we have to operate on that data if that data is not present we cannot open it. So, when that is. So, the RUN becomes 0 at that particular time and when it starts to run again it will keep on running. So, this is for within an instruction incrementing to the next; next step, next step and next step, the End signal starts a new instruction --- it means we have completely executed one instruction and now we will be moving to the next instruction. Once we have completely executed one instruction it will be fully done and then we have to again reset the step decoder.

So, the step counter basically needs to be reset. So, this is done using the End signal. So, the End signal starts a new instruction, it resets the control step counter to its starting value. So, that the next instruction can be started now. In the hardware control unit design, the sequence of operation carried out by the control unit is determined by the wiring of the logic elements, and hence it is named hardwired. Now we see that it is basically a combinational circuit; we are giving inputs and we are getting some output.

So, this basically depends on the wiring how you have put on everything in place, how you have placed the encoder, how you have use the step decoder everything, and ultimately this is the wiring. Hence it is called hardwired control unit design, and this approach of control unit design is fast, but limited to the complexity of instruction set that is implemented. So, we cannot have very complex instructions implemented using hardware, but simple instructions can be executed and it will be much fast and efficient,

but we cannot have very complex instruction implemented here. With more complex structures the complexity increases and that flexibility will go off.

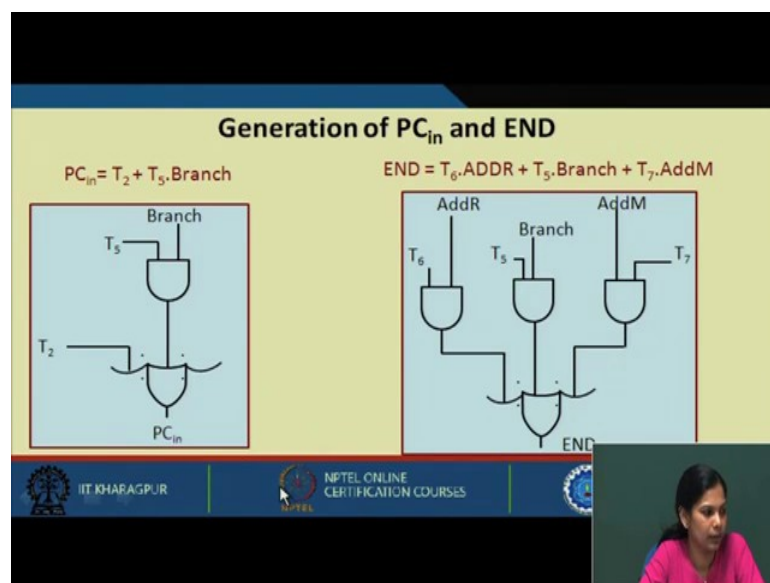
(Refer Slide Time: 16:38)

### Generation of Control Signals

	ADD R1, R2	ADD R1, LOCA	BRANCH Label
1	PC <sub>out</sub> , MAR <sub>in</sub> , Read, Select4, Add, Z <sub>in</sub>	1 PC <sub>out</sub> , MAR <sub>in</sub> , Read, Select4, Add, Z <sub>in</sub>	1 PC <sub>out</sub> , MAR <sub>in</sub> , Read, Select4, Add, Z <sub>in</sub>
2	Z <sub>out</sub> , PC <sub>in</sub> , Y <sub>in</sub> , WMFC	2 Z <sub>out</sub> , PC <sub>in</sub> , Y <sub>in</sub> , WMFC	2 Z <sub>out</sub> , PC <sub>in</sub> , Y <sub>in</sub> , WMFC
3	MDR <sub>out</sub> , IR <sub>in</sub>	3 MDR <sub>out</sub> , IR <sub>in</sub>	3 MDR <sub>out</sub> , IR <sub>in</sub>
4	R1 <sub>out</sub> , Y <sub>in</sub>	4 Address field of IR <sub>out</sub> , MAR <sub>in</sub> , Read	4 Offset-field-of-IR <sub>out</sub> , SelectY, Add, Z <sub>in</sub>
5	R2 <sub>out</sub> , SelectY, Add, Z <sub>in</sub>	5 R1 <sub>out</sub> , Y <sub>in</sub> , WMFC	5 Z <sub>out</sub> , PC <sub>in</sub> , End
6	Z <sub>out</sub> , R1 <sub>in</sub> , End	6 MDR <sub>out</sub> , SelectY, Add, Z <sub>in</sub>	
		7 Z <sub>out</sub> , R1 <sub>in</sub> , End	

So, keeping the flexibility of this hardware simple instructions are basically implemented using hardwired control unit design. Let us now see these 3 instructions together. These are the control signals required for ADD R1, R2; ADD R1,LOCA; and BRANCH Label. So, at various time steps following control signals are getting generated.

(Refer Slide Time: 17:04)





For these particular 3 instructions I will now generate the signal using hardwired control unit hardwired design for PCin and END. Now see what is PCin. If you go back we have to see where we have PCin; T1 we do not have PCin even for anyone, and in T2 we have PCin for all the instructions, and then we see that we do not have any PCin for ADD, we do not have any PCin for ADD R1,LOCA, but we have PCin in step 5 for BRANCH.

So, we can take this into consideration that PCin signal is activated at T2 for all the instructions, and PCin is activated for branch instruction in T5. So, we can write the control signal PCin as T2 for all instructions plus at T5 when it is branch instruction only. So, this logic expression can be written in the form of this logic gate where T5 and BRANCH is connected with an AND gate which is connected to the input of an OR gate.

So, this expression can be implemented using this logic function. Similarly let us see END. END happens at T6 for AddR, at T5 for Branch, and at T7 for AddM. So, we can have a logic expression at T6 for AddR, at T5 for Branch, and at T7 for AddM. So, you can see that we can implement this using the following logic expression and following logic gate where these are connected with AND gates, and this is connected finally with an OR gate. So, END signal can be generated using this particular circuit.

(Refer Slide Time: 20:30)

**Microprogrammed Control Unit Design**

- Control signals are generated by a program similar to machine language program.
- The *Control Store* (CS) stores the microroutines for all instructions of an ISA.
- The sequence of steps corresponding to the control sequence of a machine instruction is the *microroutine*.
- Each sequence of steps is a *control word* (CW) whose individual bits represent the various control signals.
- Individual control words in a microroutine are called *microinstructions*.

The diagram illustrates the data flow: IR (Instruction Register) provides input to the Starting Address Generator. The Starting Address Generator outputs to the μPC (Microprogrammed Control). The μPC also receives a Clock signal. The μPC outputs to the Control Store, which then outputs the Control Word (CW).

Now, coming to microprogrammed control unit design. In microprogrammed control unit design we have a structure like this. So, what do we have here the instruction register will give a starting address generator, and this starting address generator will hit a μPC.

So, we will have within computer a small place where we will be doing a similar kind of function like a computer. So, here the IR will provide the starting address, which will hit to the muPC. So, the muPC will hit to a memory, known as control store. So, at every clock this muPC content will hit to some very high-speed memory called control store, and based on a particular instruction provided by the IR and the starting address generator, this control store will provide a control word. And this control word is nothing but it will give the information about the control signals which will be on and off, which will be required to be activated at what time period.

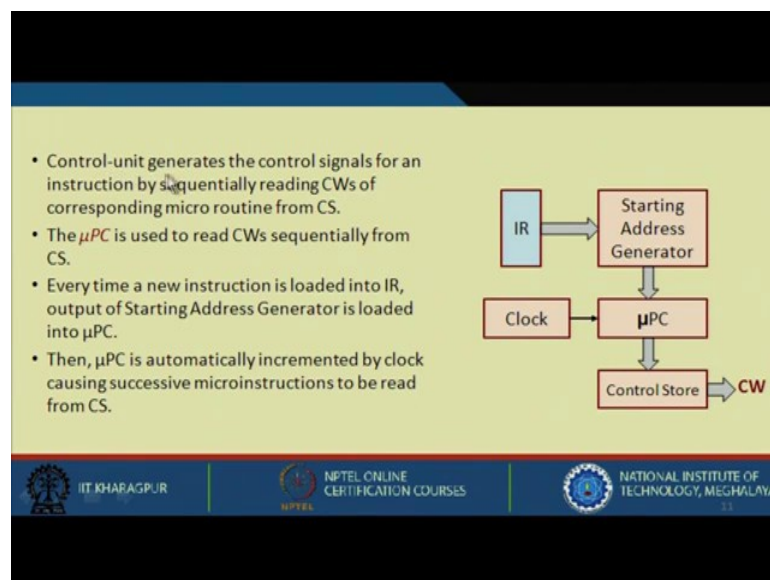
So, let us see in at this point of time we need to understand about some of the definitions. Control signals are generated by a program similar to machine language program like we were doing what we were fetching an instruction from memory, then the PC was getting incremented to the next address and again we were fetching, we were performing certain operation similar to that we will be doing something here. So, the control signals are generated by a program similar to machine language program the control store; control store is a place which stores micro-routines for all the instructions in the instruction set architecture.

So, the micro-routines are nothing but all the control signals that are required for a particular instruction. So, the control store stores the micro-routine for all the instructions of an instruction set architecture. The sequence of steps corresponding to a control sequence of a machine instruction is specified by the micro-routine. So, if an instruction requires 4 steps for execution, all the sequence of steps can be regarded as a micro routine for that particular instruction, each sequence of steps in a control word whose individual bits represent the various control signals.

So, each sequence of step is a control word; so, control word is a memory where we are storing so many things. So, let us say this is a control word 1 0 0 1 0 1 0 1 1 1 0 0 0 0 1 0 something like this we have. So, what we can say here is that, let us say this is step 1,, this is step 2 and this is step 3. So, this is basically my control store and what we are storing here? We are storing control words --- these are control words, and each sequence of steps in a control word whose individual bits represent the various control signals, meaning this is a control word and its individual bits represent some control signals. So, these are individual control signals that are on or off; that means, it is either active or not active, and individual control words in a micro-routine are called micro instructions.

Now, let us see this. So, as I said IR depending on an instruction, will generate the starting address that will hit to the  $\mu$ PC, and then  $\mu$ PC will hit the control store and from where the control words will get generated at every clock period. So, at every clock period the  $\mu$ PC will get incremented by the required amount, and then from the control store each of the control words will be generated, and the control word will give which particular control signal is required to be 1 or which particular control signal is required to be 0. So, as I have already discussed this control unit generates the control signals for an instruction by sequentially reading the control words of the corresponding micro routine from control store.

(Refer Slide Time: 26:18)



So, control store stores the control words, or we can say it stores a micro-routine. The  $\mu$ PC is used to read the control word sequentially from the control store. So,  $\mu$ PC hits the control store, and it is sequentially reads the control word one by one by one. So, every time a new instruction is loaded into IR because we when we are executing one instruction that particular starting address will be generated in  $\mu$ PC, and accordingly this will happen same way if the next instruction needs to get executed. Then the next instruction will get loaded in the  $\mu$ PC and then the starting address of that particular instruction will get loaded into the  $\mu$ PC and accordingly from the control store the control words will get generated. The  $\mu$ PC is automatically incremented by clock causing successive micro instructions to be read from control store.

(Refer Slide Time: 27:39)

**Control Store for "ADD R1, R2"**

Micro-instr.	...	PC <sub>in</sub>	PC <sub>out</sub>	MAR <sub>in</sub>	Read	MDR <sub>out</sub>	IR <sub>in</sub>	Y <sub>in</sub>	Select	Add	Z <sub>in</sub>	Z <sub>out</sub>	R1 <sub>out</sub>	R1 <sub>in</sub>	R2 <sub>out</sub>	WMFC	End	...
1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
3	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0

So, this is how it is stored. Let us say there will be many more control signals for ADD R1,R2. In step 1 what all control signals are 1, rest will be all 0. So, PCout, MARin, READ, Select4, ADD, Zin these signals will be 1, and rest will be all 0. Similarly in step 2; we need to do Zout, PCin, Yin. So, see Zout will be 1, Yin is 1, and PCin is 1 at the same time we have to wait for MFC because we have performed a read operation here in the first step. Similarly in this step these 2 signals will be activated and rest will be 0. So, this is MDRout, IRin. Similarly in step 4 for this instruction we need to perform R1out, Yin and then what we are performing we are performing R2out, SelectY, ADD and Zin. So, in Z now the result is which should be stored in R1.

So, I have to do as a Zout and R1in. So, this is a control store for ADD R1,R2. This is a micro routine for this particular instruction, and these are the micro instruction and these are stored in control store, and these are the control words that are read each time. So, the muPC will be loaded with the starting address and then at every clock period this will get incremented by 1; and it will be fetched from the control store similarly for branch we have shown. So, similarly for branch first 3 will be same.

(Refer Slide Time: 30:04)

**Control Store for "BRANCH LOCN"**

Micro-instr.	PC <sub>in</sub>	PC <sub>out</sub>	MAR <sub>in</sub>	Read	MDR <sub>out</sub>	IR <sub>in</sub>	Y <sub>in</sub>	Select	Add	Z <sub>in</sub>	Z <sub>out</sub>	IR <sub>out</sub>	WMFC	End	..
1	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0
3	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
5	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0

And for this particular thing we have to do SelectY. So, if Select4 is 1, SelectY will be 0 and we perform ADD, Zin now and then we also perform IRout because the offset field of IRout should be available, that will be added with. Y contains the previous PC value; we do perform an addADD, we do Zin and finally, we perform Zout and PCin. PC will be loaded with the new value that needs to be performed for the branch; that means, we need to go to that particular location for that branch.

So, this is how the control store looks like for branch.

(Refer Slide Time: 31:05)

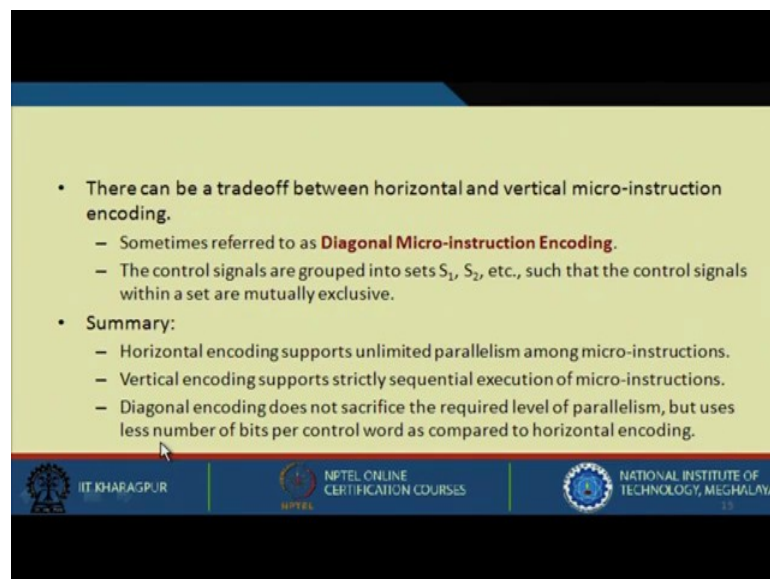
### Horizontal versus Vertical Microinstruction Encoding

- Broadly there are two alternate schemes to code the control signals in the control memory.
  - a) **Horizontal Micro-instruction Encoding**
    - Each control signal is represented by a bit in the micro-instruction.
    - Fewer control store words, with more bits per word.
  - b) **Vertical Micro-instruction Encoding**
    - Each control word represents a single micro-instruction in encoded form.
    - k-bit control word can support up to  $2^k$  micro-instructions.
    - More control store words, with fewer bits per word

Now, there are 2 alternative ways to code the control signals in the control memory. The first way is the horizontal micro instruction encoding the one I have just shown. So, each control signals is represented by a bit in the micro instruction and fewer controls store words with more bits per word is required. So, fewer control store words are required, but we require more bits per word meaning here you see that only a few signals will be 1, rest all signals many are 0; only few are 1, but we still need to keep all why because we have made it in such a fashion we are keeping for all the set of control signals. So, the size of this control word will be dependent on the total number of control signals that are present.

So, fewer control store words will be required with more bits per word, and each word will contains more number of bits compared to vertical micro instruction encoding. Here each control word represent a single microinstruction in encoded form. So, if  $k$  bit control word can support up to  $2^k$  microinstructions and more control store words with fewer bits per word.

(Refer Slide Time: 33:51)



- There can be a tradeoff between horizontal and vertical micro-instruction encoding.
  - Sometimes referred to as **Diagonal Micro-instruction Encoding**.
  - The control signals are grouped into sets  $S_1, S_2$ , etc., such that the control signals within a set are mutually exclusive.
- Summary:
  - Horizontal encoding supports unlimited parallelism among micro-instructions.
  - Vertical encoding supports strictly sequential execution of micro-instructions.
  - Diagonal encoding does not sacrifice the required level of parallelism, but uses less number of bits per control word as compared to horizontal encoding.

Let us say we have a total of 100 control signals. In case of horizontal control unit design we require 100 bits for each control word, but in this case we require only 7 bits. So, at a time only one control signals can get activated because we are encoding using just 7 bits, earlier for horizontal it was 100 bits, but for this we only require 7 bits.

So  $k$  bit control word can support up to  $2^k$  micro operation. So, if you have 100, then only 7 bits will suffice.

So, there can be tradeoff between horizontal and vertical microinstruction encoding, sometimes refer to as diagonal microinstruction encoding. So, what we do here is that the control signals are grouped into some sets and such control signals within a set of mutually exclusive. So, let us consider R1out; let us consider the single bus architecture. So, can we perform R1out and R2out together or MDRout together on any one of the operations at a time? So, we can group it into a mutually exclusive set. We require less number of bits to encode; say if there are 32 instruction, we will just require 5 bits to encode that, earlier we were using 32 bits for that.

So, the control signal are grouped into sets such that the control signals within a set are mutually exclusive. So, what is the summary out of this? So, horizontal encoding supports unlimited parallelism among microinstruction. So, many micro operations or those control signals can be activated at a time. Vertical encoding supports strictly sequential execution of micro instructions; only fewer bits are required. We require less storage, but it is sequentially performed, and it will not help for fast execution. Where the demand is for fast execution, diagonal encoding does not sacrifice the required level of parallelism, but uses less number of bits per control word as compared to horizontal encoding.

So, we can see that diagonal encoding is the best where we analyze the control signals depending on the architecture that we are using and based on that we divide into 2 groups, and we know that within that group only one operation can be performed. So, in such cases we can perform this.

(Refer Slide Time: 36:24)

**(a) Horizontal Micro-instruction Encoding**

Control Signals

- Suppose that there are  $k$  control signals:  $c_1, c_2, \dots, c_k$ .
- In horizontal encoding, every control word stored in control memory (CM) consists of  $k$  bits, one bit for every control signal.
- Several bits in a control word can be 1:
  - Parallel activation of several micro-operations in a single time step.

0 1 0 1 1 0 →  $c_2, c_4$  and  $c_5$  are activated together

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, diagonal encoding is considered as the best among horizontal and vertical. So, I will explain it in some detail. So, for horizontal micro instruction encoding, these are the control signals. Suppose there are  $k$  control signals, then in horizontal encoding every control word in the control memory consists of  $k$  bits, one bit for every control signals. So, in this case several bits in the control word can be 1; parallel activation of the several micro operation in a single time step is possible, but it is seen that in such kind of scenario where we have a very large control word many bits are 0s and few bits are 1.

(Refer Slide Time: 37:22)

- **Advantage:**
  - Unlimited parallelism is possible in the activation of the micro-operations.
- **Disadvantage:**
  - Size of the control memory is large (word size is much longer).
  - Cost of implementation is higher.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA



The advantages are that unlimited parallelism is possible in the activation of the micro operations, but the disadvantage being the large size of the control memory.

(Refer Slide Time: 37:46)

**(b) Vertical Micro-instruction Encoding**

- Again consider that there are  $k$  control signals:  $c_1, c_2, \dots, c_k$ .
- We encode the control signals in a  $m$ -bit word in the control memory, where  $k \leq 2^m$ .
- Depending on the  $m$ -bit control word, exactly one control signal will be activated (= 1), while all others will remain de-activated (= 0).
  - At most one control signal can be activated in a time step.

The diagram shows an 'm-bit word' in a box with a downward arrow pointing to a 'DECODER' box. From the bottom of the decoder box, several downward arrows point to labels  $c_1, c_2, c_3, c_4, \dots, c_k$ .

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we also need to look into this because if you want to have a control memory that is very large this cost will be much higher. So, in vertical micro instruction encoding where we see that for  $m$  bits, a decoder is used for it. So, for  $k$  control signals, we require  $k$  that is less than equals to  $2^m$ . So,  $m$  bits will be required for it depending on  $m$  bit control word exactly one control signal will be activated which is 1 while all others will be 0, but the number of bits required is much less. At most one control signal can be activated at a time. So, we cannot do PCout, MARin and all those signals at one go; we can only do first PCout then in the next step MARin, then READ, and so on.

(Refer Slide Time: 38:42)

• **Advantage:**

- Requires much smaller word size in control memory.
- Low cost of implementation.

• **Disadvantage:**

- More than one control signals cannot be activated at a time.
- Requires sequential activation of the control signals, and hence more number of time steps.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the advantage here is it requires much smaller word size in control memory, low cost of implementation. If you do not want high speed implementation you can go for such kind of thing, but when speed is a vital concern this method cannot be taken into consideration. So, the disadvantage here is more than one control signals cannot be activated at a time; requires sequential activation of the control signal and hence more number of time steps will be required here. The option is diagonal micro instruction encoding which is quite flexible.

(Refer Slide Time: 39:16)

**(c) Diagonal Micro-instruction Encoding**

The diagram shows a sequence of  $m_1$ -bit,  $m_2$ -bit,  $m_3$ -bit, ...,  $m_s$ -bit instructions. Each instruction is decoded by a corresponding decoder, which produces control signals  $c_{1,1}, c_{1,k_1}, c_{2,1}, c_{2,k_2}, \dots, c_{m,1}, c_{m,k_s}$ .

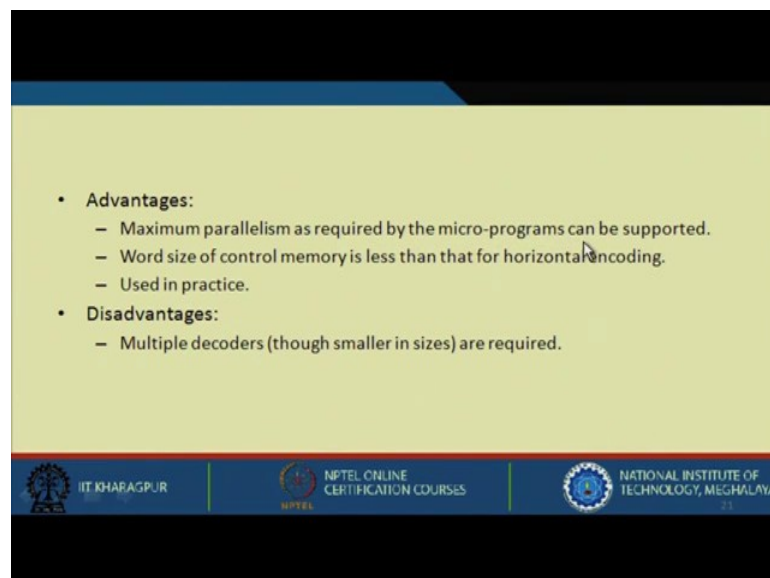
- Suppose we group the set of  $k$  control signals into  $s$  groups, containing  $k_1, k_2, \dots, k_s$  signals.
- We encode the control signals in groups as shown, where  $k_i \leq 2^{m_i}$ .
  - Within a group, at most one control signal can be activated in a time step.
  - Parallelism across groups is allowed.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, here also we require decoder. So, we group the set of  $k$  control signals into as groups containing  $k_1, k_2, k_3$ , etc. So, this group is having mutually exclusive instruction, this group is having mutually exclusive instruction, and we know that at a time only one signal from this group will get activated.

So, we encode the control signals in group as shown where  $k_i$  is less than equals to  $2^{m_i}$  this is depending on the size. So, within a group at most one control signal can be activated in a time step; parallelism across groups is allowed; also the number of bits required is less. So, by studying the architecture well in advance we can design such kind of microinstruction encoding to design a control unit.

(Refer Slide Time: 40:29)



The slide contains the following text:

- **Advantages:**
  - Maximum parallelism as required by the micro-programs can be supported.
  - Word size of control memory is less than that for horizontal encoding.
  - Used in practice.
- **Disadvantages:**
  - Multiple decoders (though smaller in sizes) are required.

The slide footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

So, the advantage here is maximum parallelism as required by the micro program can be supported and the word size of the control memory is less than the for horizontal encoding and this is used in practice. This is basically what is used in practice because parallelism can be exploited as well as the space is minimized, multiple encoders those smaller in sizes are required. So, that will take up some space, some area let us take an example suppose there are 100 control signals in a processor data path.

(Refer Slide Time: 41:00)

**Example 1**

- Suppose there are 100 control signals in a processor data path.
  - a) For horizontal encoding, control word size = 100 bits.
  - b) For vertical encoding, control word size =  $\lceil \log_2 100 \rceil = 7$  bits.
  - c) For diagonal encoding, suppose after analysis of the micro-programs, we divide the control signals into 5 groups, containing 25, 15, 40, 5 and 15 control signals respectively.
    - We have:  $m_1 = 5, m_2 = 4, m_3 = 6, m_4 = 3, m_5 = 4$
    - Control word size =  $5 + 4 + 6 + 3 + 4 = 22$  bits.

$25 \leq 2^5$	$15 \leq 2^4$
$40 \leq 2^6$	$5 \leq 2^3$
$15 \leq 2^4$	

Logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA are visible at the bottom of the slide.

So, for horizontal encoding control word size will be 100 bits. So, each will be 100 bits for vertical encoding  $\log_2 100 = 7$  bits will be required and for diagonal encoding suppose after analysis. So, it has been generated that the groups the following groups can be made that is the following groups of mutually exclusive signals can be made that is 25, 15, 40, 5 and 15.

So, in each of the cases let us understand how many bits will be required and what will be the size of the decoder that is required. So, we have  $m_1$  first group 25. So,  $2^5$  that is 25 is less than 32. So, 5 bits similarly this is 15 which is less than  $2^4$ . So, it will be 4 this is less than  $2^6$ . So, it will be 6 this will be  $2^3$  less than that. So, it will be 3 and this will be 4. So, the control word size will be  $5 + 4 + 6 + 3 + 4$  which is 22. So, we if we just need 7 bits it will be much slower we cannot afford to have this, and if we take 100 bits the space is too large. So, this is quite a feasible thing that we can take up, that is horizontal that is diagonal encoding where we exploit both the features.

So, we came to the end of lecture 20 where we have discussed about the various ways we can generate control signals and by this we have also come to the end of control unit design, but in next 2 lectures, we will be looking particularly how the control unit of MIPS is designed.

Thank you.