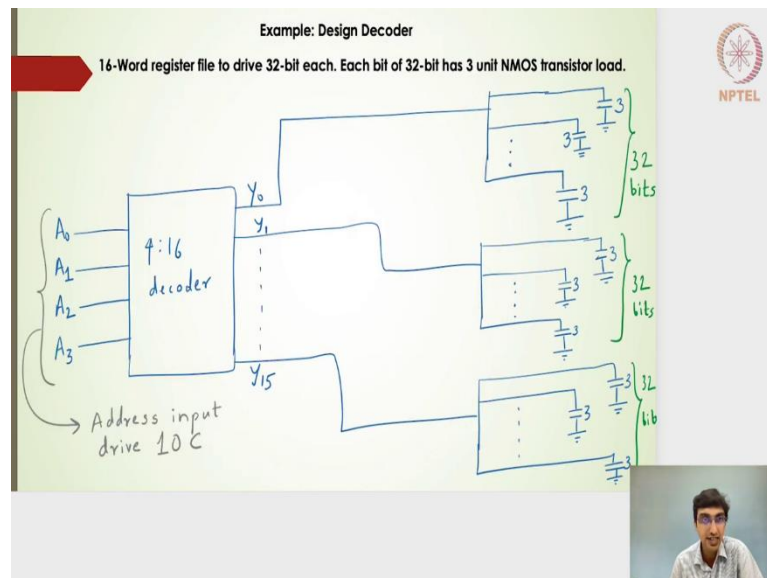


Design and Analysis of VLSI Subsystems
Dr, Madhav Rao
Department of Electronics and Communication Engineering
Indian Institute of Information Technology, Bangalore

Lecture - 38
Decoder Design

Hello students welcome to this lecture on 4:16 decoder design. We will be designing a 4:16 decoder by optimizing the sizes and optimizing the number of stages and also what kind of gates should go into that particular decoder design, that we eventually get a lowest or the minimum delay.

(Refer Slide Time: 00:44)



This is an example I have just taken it from the western Harris textbook and one can read the whole description of this particular exercise or an example that is given in the textbook. Just to simplify I have drawn a block diagram of 4:16 decoder, 4 inputs and the 16 outputs and I have returned it in the form of A_0, A_1, A_2, A_3 as the inputs and the 16 outputs being y_0 to y_{15} .

In that particular example, in the description it has been given that there is a 16 word register file to drive the 32 bit each. What it means is, this particular 16 output bits which is generated from the decoder its actually used to design the 16 word register file and each of this outputs it goes into individual bits.

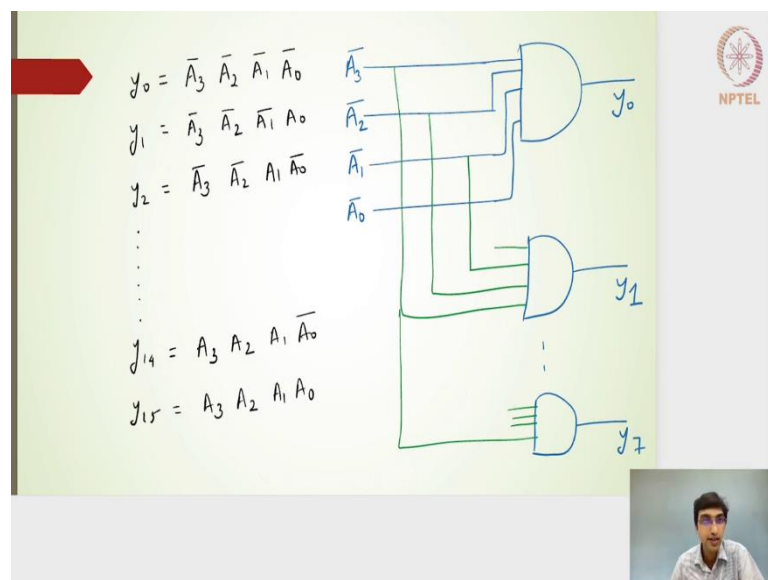
This particular output is driving that another 32 bit system and each of this bit sees a capacitance of 3C. We have the 16 output bits coming out from the decoder and each of these output bits sees a 32 bit system, each of the 32 bit is estimated to have a load capacitance of 3C, where a 1C indicates the unit NMOS capacitance.

3 of this 3C represents 3 times the unit NMOS capacitance and we have 32 bits. Each of this output coming from the decoder or I can consider the 16 word register file, where one register file is seeing a load of 32 x 3C, around 96C. Each of these output bits sees a load capacitance of 96C.

It is also given that particular exercise is the inputs driving capacitance of 10C. What it means is if this input is connected to an inverter or to a NAND gate or to a NOR gate and whatever we have that first stage it is accommodating a 10C capacitance and then we have the load capacitance of 32 x 3 which is around 96C.

With this particular configuration we are supposed to design the best number of stages for this 4:16 decoder block, the best sizes that we will get the minimum delay, hope the problem statement is kind of clear, I just simplified this particular problem statement and then given it to you.

(Refer Slide Time: 03:44)



Let us begin with the decoder design. What we have studied in the digital design course the decoder design can be expressed as the product of all the inputs. We have y_0 to y_{15} , y_0

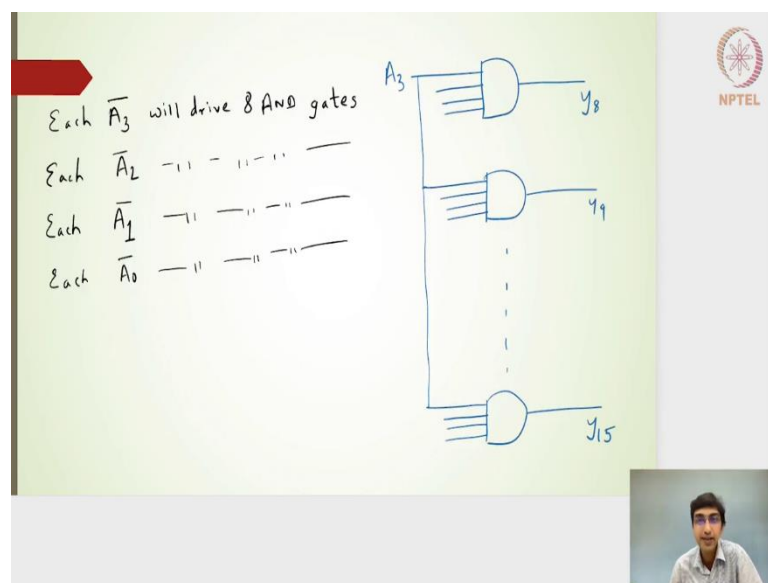
all the inputs being 0 we will get y_0 as the output y_1 , if all the 3 inputs are 0 and then one of the lowest significant bit if it is 1 we will get the y_1 and similarly y_2, y_8 or y_7, y_8 and similarly y_{14} and y_{15} , y_{15} represents where all the inputs are one we will get the y_{15} bit.

When I have drawn this particular schematic a gate level schematic for y_0 to y_7 and in the next slide I have the schematic for y_8 to y_{15} . In this particular $y_0 = \overline{A_3} \overline{A_2} \overline{A_1} \overline{A_0}$ connected to this AND gate and then we will get this y_0 and similarly $y_1 = \overline{A_3} \overline{A_2} \overline{A_1} A_0$ and not take it from the $\overline{A_0}$. That is a slight difference here between y_0 and y_1 . This one being A_0 and not $\overline{A_0}$ and similarly we will have the $y_7 = A_2 A_1 A_0 \overline{A_3}$. If I consider the $\overline{A_3}$ input that is going to generate y_0 that is going to generate y_1 that is going to generate y_2, y_3, y_4, y_5, y_6 and y_7 .

This $\overline{A_3}$ line is actually going to 8 such output logic generation, to 8 such gates. So, as to generate 8 output logics of the decoder block, that means that $\overline{A_3}$ is actually having an 8 branches. It is having an 8 branches that is going and feeding into the next gate.

As to produce the 8 output signals kind of very important to incorporate this branching factor into our design. Similarly if I notice $\overline{A_3}$ is getting branched 8 times I am sure that A_3 we will also get branched 8 times starting from a y_8 to y_{15} output generation.

(Refer Slide Time: 06:38)

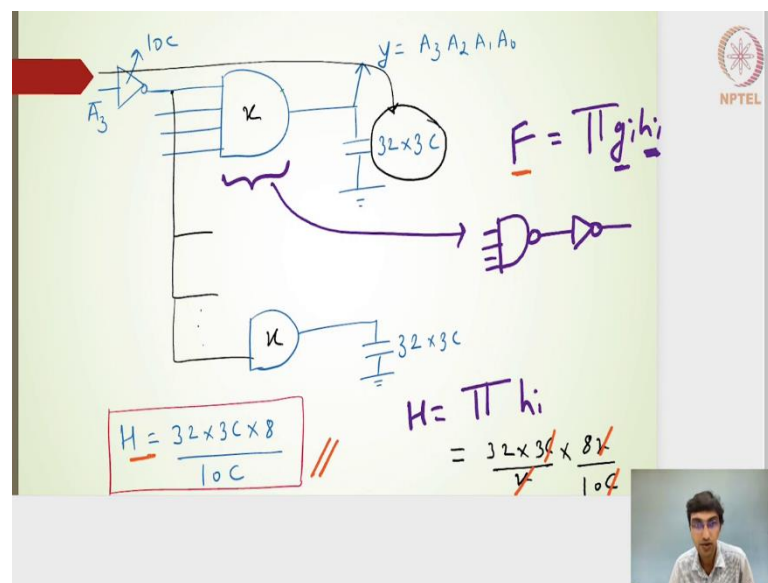


Let me go to the next slide. This is what the schematic says A_3 is going now to 8 such NAND gates to generate the 8 outputs of the decoder block y_8 to y_{15} . A_3 is also getting

branched 8 times. In fact, we can have a closer look at all the output logics and we can say that A_3 will drive 8 NAND gates, $\overline{A_3}$ will drive 8 AND gates, A_3 will drive 8 AND gates, $\overline{A_2}$ will drive 8 AND and then similarly $\overline{A_1}$, $\overline{A_0}$, A_1 and A_0 all of them will be driving 8 NAND gates.

If I am considering A_3 or if I am considering $\overline{A_3}$ or any of the inputs and its complement, I can say that each of these input is having a branching factor of 8. It means that it is going to get branched 8 times to generate the 16 outputs, hope this is clear.

(Refer Slide Time: 07:43)



Moving forward, let me take a very primitive design stage, a primitive step towards building the optimized or the most optimized 4:16 decoder. Let us say that this is my $\overline{A_3}$ and then an inverter which sees the driving strength or the driving capacitance of $10C$ that means, that this inverter will have some size. The input of $\overline{A_3}$ has a capacitance of $10C$ and that is something given to us in the design stage, this is the output of this particular inverter is going to the next stage of the AND gates.

Now as we have clearly seen A_3 is getting branched 8 times. I have drawn this the branches kind of a dotted lines I have not completed this, but let us assume that this A_3 goes to 8 such AND gates, the output of this AND gates is going to produce the output of the decoder and then I have stated this particular logical expression for one of this output of the decoder which is $A_3 A_2 A_1 A_0 = y_{15}$.

A_3 going to the AND gates and then generating the y_{15} . In fact, A_3 will go to the other 7 AND gates and then it will generate y_{15} , y_{14} , y_{13} , y_{12} , y_{11} , y_{10} , y_9 and then finally y_8 . Basically, A_3 is getting branched 8 times to get the outputs of starting from y_8 to y_{15} and each of this output sees a load capacitance of 32 into multiplied by $3C$ the input is anyways $10C$.

I will start with $\overline{A_3}$ have an inverter and then we will have the A_3 . Similarly we will have to generate A_2 it will come along with an inverter and then the input to that particular inverter will be nothing but $\overline{A_2}$ and then similarly if I want to generate $\overline{A_3}$ which will get branched to 8 AND gates. So, as to generate y_0 to y_7 , here I need $\overline{A_3}$, that means the input to this inverter will be A_3 .

That will be my structure and in this particular structure what we are seeing is we are having not only the 4 inputs, but also the 4 complementary inputs as a requirement. Let us say if I go back to this slide number 2 and the example or the description says that not only the 4 inputs are fine, but its complement is also perfectly fine, while you are considering or while you are designing this 4:16 decoder.

Basically, we are considering 8 inputs, 4 of the A_0 to A_3 and then 4 of this complementary inputs and at the output we are generating y_0 to y_{15} and it is not necessary that it could be an active low or an active high output. That is also being stated in this particular specification we could generate a $\overline{y_0}$, $\overline{y_1}$, $\overline{y_{15}}$ or we can generate y_0 or y_1 , y_{15} , either active low signals at the output or active high signals.

But the only one specification is we have to generate output either active low consistently for all the 16 outputs or active high signal consistently for all the 16 outputs, it should not be like y_0 will be an active low signal and y_1 will be an active high signal and y_2 will be an active low signal and so on. We have to maintain that consistency throughout the 16 output bits.

Going back to the present slide, this is what we have. We have 4 of the input and its 4 of its complementary input. We have total 8 inputs getting branched 8 times each of the input is grading branch 8 times generating an output starting from y_0 to y_{15} , each of the output sees a load capacitance of $32 \times 3C$ which is nothing but $96C$.

The overall gh, what we need is the path effort and I can calculate the path effort,

$$F = \prod g_i h_i$$

In this case we have an inverter here and then we have an AND gate. In the CMOS AND gate can be represented by nothing but the 4 input NAND gate followed by an inverter. It is a very simple form of representing the AND gate. If I consider that I can find out what is the g could be the multiplication of this particular inverter, g the 4 input NAND gate, g the logical effort and then the inverters logical effort.

What should be the multiplication of all individual stages electrical effort which will be nothing but and I have stated here H is nothing but I can consider it to be the multiplication of all H is which will be nothing but a total capacitance of $32 \times 3C$, which is $96C$ and there is a branching of 8,

$$H = \prod h_i = \frac{32 \times 3C}{x} \times \frac{8x}{10C}$$

$$H = \frac{32 \times 3C \times 8}{10C}$$

What we have done is we have understood the problem statement and we realize the problem statement in terms of a very simple design just to mention that this is not the final design this is just a simple design.

We are trying to evaluate eventually if we want to evaluate what is F the path effort and once we have the path effort then we are going to have the best stage effort and then we will realize a higher level design. In this particular process what we have identified is the capital H .

(Refer Slide Time: 15:23)

NPTEL

$G = \prod g_i = 2$

$F = 2 \times H = \frac{2 \times 32 \times 8 \times 3}{10} = 153.5$

3-stages $\Rightarrow \hat{F} = F^{1/3} = (153.5)^{1/3} = 5.26$
for INV-NAND-INV

Delay = $3 \times 5.26 + 6 = 21.79$

Between 2.4 and 6!!
But can we get stage-effort closer to 3.59?

Moving on, this is what we have realized an AND gate can be represented in the form of a 4 input NAND gate followed by an inverter we know its logical effort is 2, the logical effort of the inverter is 1. The overall path effort for an inverter followed by a 4 input NAND gate followed by an inverter will be nothing but with the logical effort is nothing but $g = 1$ here for an inverter. The first stage is nothing but an inverter which goes into the NAND gate which goes into an inverter. We will see the logical effort,

$$G = \prod g_i = 2$$

$$F = 2 \times H = \frac{2 \times 32 \times 8 \times 3}{10} = 153.5$$

Here we have used 3 stages, my individual best stage effort will be nothing but,

$$\hat{f} = F^{1/3} = 153.5^{1/3} = 5.26$$

In our overall design rule we have stated that anything between 2.4 to 6 for a general circuit it is appropriately fine because we are going to get a delay which is 15 percent tolerable, which is 15 percent more than the best design, 5.26 is going to it is kind of falls in between range of 2.4 and 6. If I design the 5.26 individual stage effort, my delay should be kind of acceptable.

But the question is can we get the stage effort closer to 3.59, can we improve our design in such a way that the delay is not only 15 percent, within the 15 percent range of the best design, but also very very close to the best design for this particular circuit, the 3 stage which is nothing but an inverter which is kind of branched 8 times and then followed by the 4 input NAND gate followed by an inverter here.

The overall delay for this particular 3 state circuit will be nothing but,

$$\text{Delay} = 3 \times 5.26 + 6 = 21.79$$

If I want to find out the absolute delay it will be nothing but $21.79 \times 3RC$ and if rc turns out to be 1ps it will be $21.79 \times 3ps$, hope the very first step to get into the 4:16 decoder design is clear.

(Refer Slide Time: 18:46)

Ideally $\hat{f} = 3.59$, $F = 153.5$
 $\hat{N} = \frac{\ln(F)}{\ln(\hat{f})} = 3.93 \approx 4 = N$ $\hat{N} = 3.93$
 If $N = 4$, $\hat{f} = (153.5)^{1/4} = 3.51$
 delay = $4 \times 3.51 + 7 = 14.04 + 7 = 21.04$ inv

But what we really want is an $\hat{f} = 3.59$ to get closest to the best delay and my capital $F = 153.5$. What should be the best stage,

$$\hat{N} = \frac{\ln(F)}{\ln(\hat{f})} = 3.93 \approx 4$$

If $N=4$,

$$\hat{f} = 153.5^{1/4} = 3.51$$

In that sense if I choose the number of stages to be 4 and I can easily do it with this particular design, because I have the 4 input NAND gate followed by inverter and before that we had an inverter, I will follow it with one more stage of an inverter. This is an additional one extra inverter to our last circuit design. Let me start with $\overline{A_3}$ here at the input side because we are perfectly fine with the complementary inputs that what is up that is something the problem statement specify.

Then we have this inverter which is giving me A_3 goes to the 4 input NAND gate and then I will get some particular expression here, of course here it is coming from A_2, A_1 and A_0 which is also coming from its respect to inverter output. The output of the NAND gate which is to go to the inverter and then it will give me A_3, A_2, A_1, A_0 and then additional inverter just to ensure that we are having the best delay or closest to the best delay.

Then the output of this will be nothing but A_3, A_2, A_1, A_0 and then the bar of that, this represents an active low output, $32 \times 3C$ is its load capacitance alright. This is our design this is now our second circuit remember that the first circuit had inverter followed by the 4 input NAND gate followed by the inverter this one we have added one more inverter here or one extra inverter here.

That we will get closest to the stage effort will be closest to 3.93 or rather the stage effort will be close to 3.59. In that sense I will have 4 stages, if I have 4 stages the $\hat{f} = 3.51$ which is very very close to 3.59 which is very good, the overall delay here you can calculate, it will be nothing but 4×3.51 plus 1 inverter parasitic of 1, 1 inverter parasitic of 1, 1 inverter parasitic of 1 and then this parasitic of 4, it will be 7. It will be nothing but if I calculate this it will turn out to be,

$$\text{delay} = 4 \times 3.51 + 7 = 14.04 + 7 = 21.04$$

If I take the previous case where I had 3 stages, I had the delay of 21.79 and now in the current circuit design I have 21.04 as a delay. hope this is clear at this particular point of time. What we had seen is the 2 circuits which we have arrived with our primitive expressions of the decoder outputs and we have seen two circuits, one circuit is giving 21.79 as the normalized delay and another one is 21.04.

(Refer Slide Time: 22:57)

Circuit#1 : N=4, and one 4-Nand gate

$$\hat{D}(N=4) = 1 + 4 + 1 + 1 + 4f = 7 + 4(3.51)$$

$$\hat{D}(N=4) = \underline{21.04}$$

Moving on, this is our previous circuit. We have represented this sizes for input NAND gate as X inverter of Y and inverter Z this particular inverter has a size of 10 and this is what we have arrived at 21.04, this is our circuit one with $N = 4$ stages. I am considering this as circuit 1 because circuit number 0 which was previous to this had 3 stages inverter followed by the 4 input NAND gate followed by an inverter. That used to give us 21.79. I am not going to use that because we have evolved this particular circuit from that particular circuit. This seems to be the best circuit at this particular point of time, comparing both the circuit 1 turns out to be the better circuit for $N = 4$ stages.

(Refer Slide Time: 23:53)

Circuit#2 : N=4, and two 2-Nand gates

4 branches

2 branches

$$F = GH = 1 \cdot \left(\frac{4}{3}\right)^2 \cdot 4 \cdot 2 \cdot \frac{32 \times 3}{10} = 136.46$$

$\frac{4}{14}$
 $\frac{4}{13}$
 $\frac{4}{12}$

Moving on, this is another circuit here, circuit number 2 with $N = 4$ stages. Now I have got a hang of 4 stages. Why not make some changes to the 4 input NAND gates and recreate that output logic, why do you have unnecessarily 4 input NAND gates which sees a logical effort and the parasitic to be larger. The 4 input NAND gates had a parasitic of 4 a logical effort of almost $2N + 2/3$, which will be $4 + 2, 6 / 3$ the logical effort is 2.

Why not I can improve, I mean the question is can we further refine the 4 stages into a smaller number of NAND gates. I have now got 2 such 2 input NAND gates, instead of 4 input NAND gates, I am creating 2 such 2 input NAND gates here, the 2 input NAND gates requires me to add one more 2 input NAND gate in the path, as to get an output of the active low output signal.

This is basically the active low y_{15} output, there is a slight modification in a design $\overline{A_3}$ goes to the inverter very similar here it is, now instead of the 4 input NAND gate it is now 2 input NAND gates again $\overline{A_3 A_2}$ will be generated and then it goes to the inverter producing A_3, A_2 and then goes to the NAND gate giving us the active low y_{15} output.

Active low y_{15} output, A_3 here gets branched to another two input NAND gate which is size of X again, we need a same size here and again the same size here of Y of course, it produces $\overline{\overline{A_3 A_2}}$. That it should be able to create not only this 4, the other 4 output signals, A_3 gets branched 2 times to generate the other 4 output signals and similarly we will have $\overline{A_3}$. This $\overline{A_3}$ is actually generating this A_3 signal goes to the 4 goes to the 2 branches to generate the 8 outputs, similarly here if I consider the counterpart of A_3 going to an inverter and then generating $\overline{A_3}$, now it will go to 2 branches and generate the remaining 8 output logics.

In this particular path very interesting to see in this particular path of $\overline{A_3}$ to y_{15} here it will get branched twice and then somewhere here once A_3 and A_2 are generated it is going to get branched 4 number of times. 4 times it will get branched one to generate the y_{15} , the other one to generate the y_{14} , the other one to generate y_{13} and the other one to generate y_{12} .

This design is kind of very very homogeneous in that sense in every path from the input to the output. In this particular path or every other path it gets branched in 2 points or in two intermediate points. One at this particular level it will get branched twice and then the

another one it will get branched after this particular inverter state it will get a branch 4 times.

The overall branching factor is still 8, if I want to find out the path effort for this particular $N = 4$ stages turns out to be nothing but the logical effort is 1 here, the logical effort is $4/3$ here, $4/3$ here, the logical effort is 1 here. It will be $4/3$ the whole square and the overall product of the electrical effort of all these 4 stages turns out to be $(32 \times 3C)/Z$ and then multiplied by $4Z/Y$.

$32 \times 3 \times 4$ will come and then we will have this stages which will be nothing but Y/X and here for this particular stage it will be $2X/10$. The branching factor of 2 here and a branching factor of 4 here will come, which will get multiplied by $(32 \times 3C)/10$. That is what we have $(4 \times 2 \times 32 \times 3) / 10 = 136.46$, it is kind of very important that we started with F value which is around 153 and derived the 4 stages and then when I am trying to modify this particular design with the help of 2 input NAND gates, because 4 input NAND gates has a greater parasitic and a greater logical effort. Why not modify that with A_2 input NAND gates turns out that the path effort is reducing, which is in a way it is good because the reduced path effort is likely to give me a reduce F cap and the overall delay will be less.

(Refer Slide Time: 29:43)

$F = 136.46$
 $\hat{F} = F^{1/4} = 3.41$
 $\hat{D}(N=4) = \frac{1}{2} + \frac{2}{2} + \frac{1}{2} + \frac{2}{2} + 4\hat{f}$
 $= 6 + 4(3.41)$
 $\hat{D}(N=4) = 19.64$
Inverter-2Nand-Inverter-2Nand is better than Inverter-4Nand-Inverter-Inverter.
 21.09

Let us have a look, if the path effort is,

$$F = 136.46$$

$$\hat{f} = F^{1/4} = 3.41$$


The overall delay here 2 input NAND gates and then 2 inverters will give me,

$$\begin{aligned} \hat{D}(N = 4) &= 1 + 2 + 1 + 2 + 4\hat{f} \\ &= 6 + 4(3.41) \end{aligned}$$


$$\hat{D}(N = 4) = 19.64$$

Inverter with A₂ input NAND gate followed by an inverter which is followed by 2 input NAND gate is better than the inverter followed by the 4 input NAND gate followed by inverter and then the inverter because it was giving me a somewhere around 21.04 whereas, this one is giving 19.64. This turns out to be a much closer delay value. We will prefer this particular circuit over any other circuits, hope this is clear.

(Refer Slide Time: 30:45)



Design	Stages <i>N</i>	<i>G</i>	<i>P</i>	<i>D</i>
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	3	2	6	22.1
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6



Moving forward, this is what one can actually get different designs, 2 stage designs, 3 stage designs, 4 stage designs and it turns out that the 19.64 or 19.7, whichever we have got that turns out to be the best design. 19.7 is our best design compared to the other variants.

(Refer Slide Time: 31:15)

Steps for designing an optimized digital circuit

1. Find the Logic.
2. Determine the F using simple logic, and also accommodate the number of branches.
3. Determine the number of stages using the calculated F using $N = \ln(F)/\ln(3.59)$.
4. Determine the delay using N stages.
5. Try modifying the design using N stages and determine new F , and new N for minimum delay.
6. Compare all possible designs and find the best.

This is kind of a general rule one can follow. If there is a particular example of a decoder design or it could be in some kind of a different logical output expression that is given to us. So as to do an optimized design for the minimum delay. The first step is to find the logic here we have identified the logic of the decoder a very simplified expression.

We could also realize that using the gate level, we took an AND gate and determine the F . That the overall path effort we could define basically here we need to identify what is the number of branches the output expressions for the different output expressions that will take. That the inputs at some stages it will get branched and if we can identify that we should be able to find out what is the F value or the path effort value.

Once we have the F value we can identify what is the best stages with respect to our generalized rule of 3.59. Individual stages seeing a 3.59 if we can get an N value which is closest to the \hat{N} value. We cannot always have an \hat{N} value which will give me 3.59 as the stage effort, but if you can calculate the \hat{N} value and then we can find out what is N which is closest to the \hat{N} value.

Then we should be close to our optimized design and once we have the N value which is nothing but in this particular case we identified 4 as the stages, we determine the delay with the 4 stages and then we try modifying the design. Instead of the 4 input NAND gate we adopted the 2 input NAND gates and then we adopted the 2 input NAND gates in the final stage also, so that we will get the output expression to be that of what we expect.

Using the N stages determine the new F . The new F turned out to be less than what we had calculated in this particular step instead of 153.15 it was now 136 and we will be able to evaluate the new N for the minimum delay and then calculate the minimum delay.

Similarly, we can keep on reiterating this step 5 to 3 and then go down to 5 and then keep on doing it and we can form a table and then we should be able to compare all the possible designs and then find the best one. This is the process of getting the critical path delay for any kind of an digital output circuit.