Circuit and Layout Design for Optimum Performance

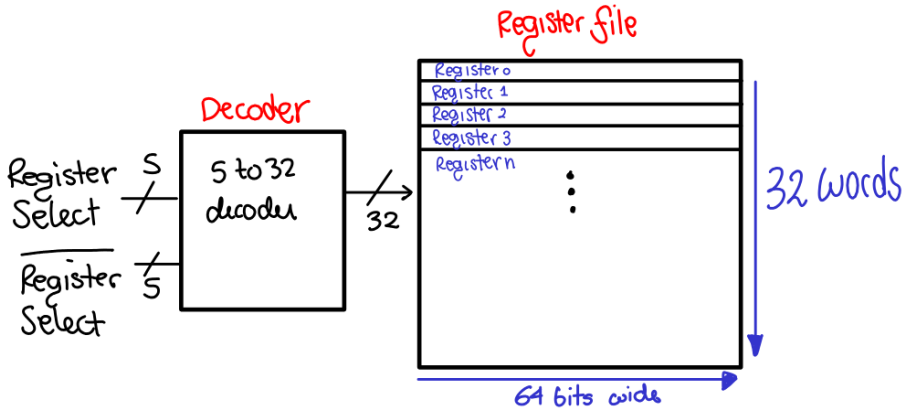
1. Circuit characterization and performance estimation

1.1 Investigate the design of a decoder for a 32-word register file where each word line is intended to drive 64-bit registers. (Unit transistor of 4λ/2λ: λ = 0.2 μm)

For this question we are asked to devise 6 different row decoder designs (following a similar procedure to that outlined in section 4.5.3 from the reference textbook), with the following requirements:

- Row decoder intended to drive 32 word register file
- Do not include registers, but consider their presence for capacitive loading
- Each word-line select should be able to drive 64 bit registers
  - *Each register bit presents a load of 3 unit-sized transistors on the world line (our loading)*
- True and complementary versions of the address bits adx[4:0] are available
- Each address input from the previous stage can drive up to 10 unit-sized transistors
- Only Inverter and NAND gates are available for your design
- Explore number of stages: 2-6, as you estimate path performance
- Explore NAND gates with inputs: 2-5, as you estimate path performance

The following is the general view of the decoder needed for a 32-words register file with 64 bits-wide words.



To calculate the logical effort for a row decoder path, one needs to know the optimal number of stages which is also an unknown (it's a cyclical problem) . The book proposes two starting points for determining the optimal number of stages: (1) is to start from a given arbitrary number of stages and determine the path logical effort which then we can use to calculate the optimal number of stages, the other (2) is to assume (initially) that the path logical effort G=1 (for a simple decoder) and then revise our results to arrive at the optimal number of stages. Both approaches will be explored below:

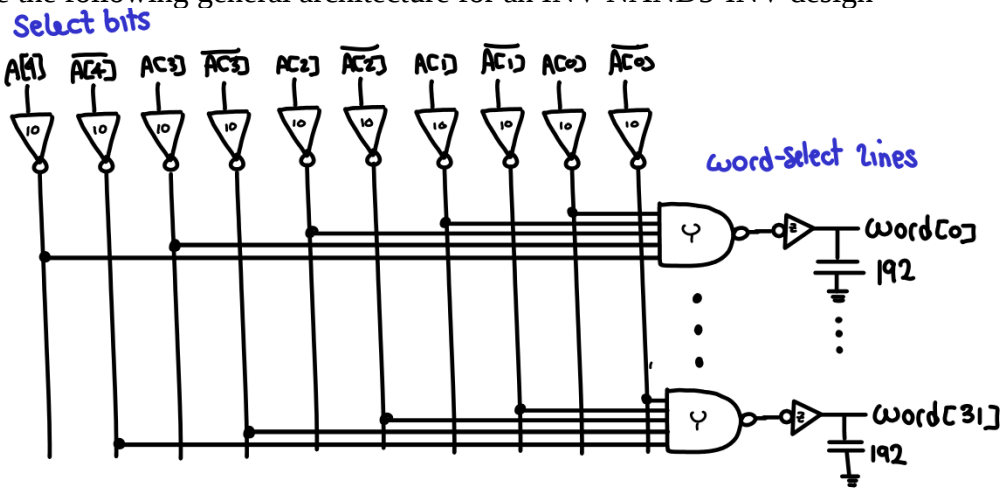
In our case, taking a starting arbitrary 3-stage decoder (to use the book example as a starting reference), we have: for a 32 word decoder we will need 2^N gates with each gate having N inputs. (section 12.2.2)

3 Stage Row Decoder: INV-NAND5-INV

Then, our initial starting 3-stage decoder:

- INV-NAND5-INV design
- 32 NAND5 gates (5 input NAND gates)

And will have the following general architecture for an INV-NAND5-INV design



Note here that the output load on each word line is 64 bits (register width is 64 bits) with 3 units of capacitance each (i.e. 192 units of capacitance for each row select line), and we know from before that each address line can drive 10 unit-sized transistors: $C_{\text{in\_path}} = 10\,units \quad C_{\text{out\_path}} = 192\,units$

This then determines our **path electrical effort H:** $H = \frac{C_{\text{out\_path}}}{C_{\text{in\_pat}}} = \frac{192}{10} = 19.2$ This path electrical effort will be common for any decoder design we choose – the input and output loading will be the same regardless.

To calculate the branching effort, we can notice that for every address input a[i] after passing the inverter we have 16 branching options (b1 = 16) and after passing our NAND gate we have only 1 branching option (b2=1), therefore our **path branching effort B:** $B = \Pi\, b_i = b_1 x\, b_2 = 16$

Now looking at one individual path, we can calculate the **path logical effort G:**

$G = g_{inv}\, x\, g_{nand5}\, x\, g_{inv} = 1\, x\, \frac{(5+2)}{3}\, x\, 1 = 7/3$ and subsequently we can calculate the actual path effort for our INV-NAND5-INV 3-stage decoder, **path effort F:** $F = GBH = 7/3\, x\, 16\, x\, 19.2 = 716.8$

Then, to optimize for minimum path delay (delay is smallest when each stage bears the same effort), each stage effort should be distributed equally, **optimal stage effort f^**

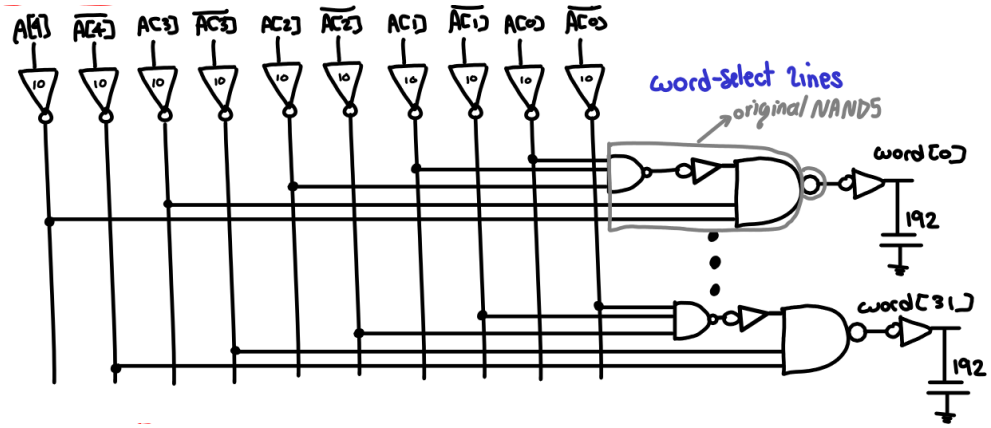
$f^{\wedge} = gh = F^{(1/N)} = \sqrt[3]{F} = \sqrt[3]{716.8} = 8.95$ Here we see that unfortunately our stage effort does not fall within our desirable 2.4 → 6 range and is then most likely not optimal for performance (significant cost in speed)

We can calculate our **path parasitic delay P as:** $P = \Sigma p_i = p_{inv} + p_{nand5} + p_{inv} = 1+5+1 = 7$ and therefore our **total path delay D is:** $D = NF^{(1/N)} + P = N f^{\wedge} + P = 3\, x\, 8.949 + 7 = 33.847$

At this point, with our known path effort calculated above we can estimate the **optimal number of stages N^:** (following procedure on page 169 on reference textbook) $N^{\wedge} = \log_4(F) = 4.743 \sim 5$

Hence, we know a 5-stage decoder most likely will exhibit superior performance as compared to other decoder designs with different number of stages. Subsequently we will estimate total path delay for our 5 stage row decoder, and finally we will estimate total path delays for other row decoders. (for reference only)

5-Stage Row Decoder: INV-NAND3-IN-NAND3-INV



For our 5-stage row decoder, the process to estimate and arrive at the total path delay is the same as before, mainly:

Path logical effort G: $G = g_{inv}\,x\,g_{nand3}\,x\,g_{inv}\,x\,g_{nand3}\,x\,g_{inv} = 5/3\,x\,5/3 = 25/9$

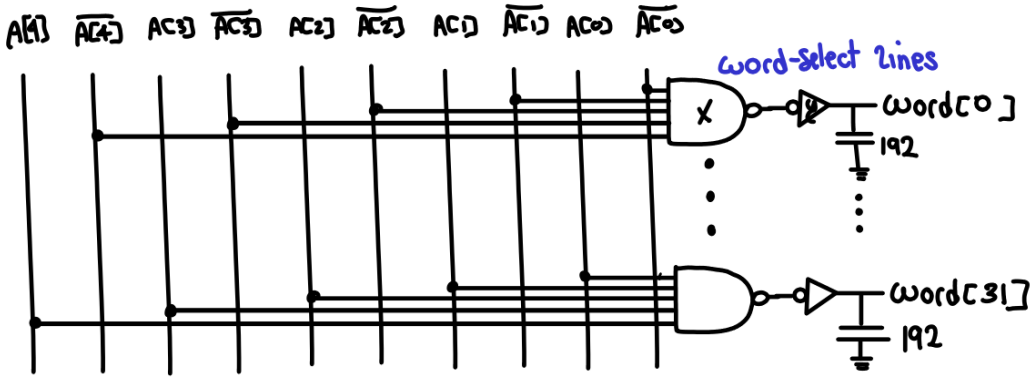
Path Effort F: $F = GBH = 25/9\,x\,16\,x\,19.2 = 853.33$

Optimal stage effort f^: $f^{\wedge} = gh = F^{(1/N)} = \sqrt[5]{F} = \sqrt[5]{853.33} = 3.857$

Path Parasitic Delay: $P = \Sigma p_i = p_{inv} + p_{nand3} + p_{inv} + p_{nand3} + p_{inv} = 1+3+1+3+1 = 9$

Total path delay D is: $D = NF^{(1/N)} + P = N f^{\wedge} + P = 5\,x\,3.857 + 9 = 28.284$

2 Stage Row Decoder: NAND5-INV



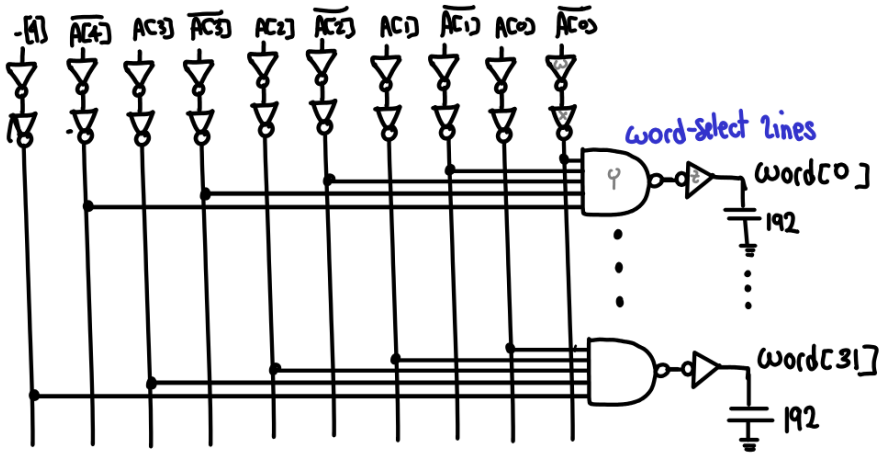
Following the same process as before (not repeated here for conciseness)

Path logical effort G: $G = 7/3$ **Path Effort F:** $F = 716.8$

Optimal stage effort f^: $f^{\wedge} = 26.773$ (too large, indicates not optimal number of stages)

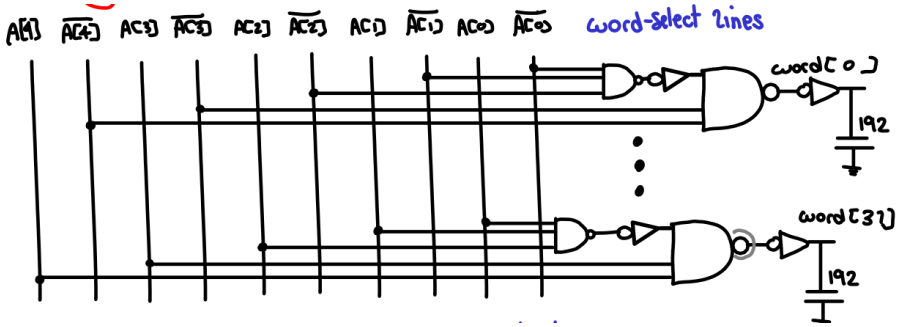
Path Parasitic Delay: $P = 6$ **Total path delay D is:** $D = 59.546$

4 Stage Row Decoder (4a): INV-INV-NAND5-INV



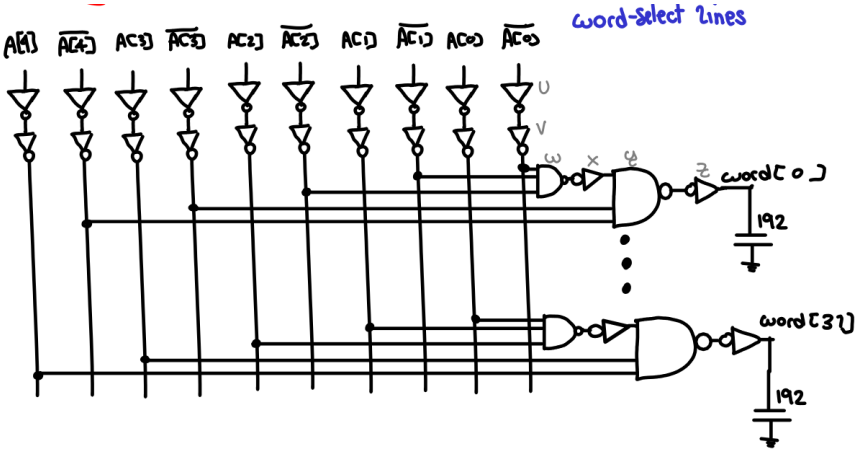
Path logical effort G: $G=7/3$ **Path Effort F:** $F=716.8$ **Optimal stage effort f^:** $f^{\wedge}=5.174$
Path Parasitic Delay: $P=8$ **Total path delay D is:** $D=28.697$

4 Stage Row Decoder (4b): NAND3-INV-NAND3-INV



Path logical effort G: $G=25/9$ **Path Effort F:** $F=853.33$ **Optimal stage effort f^:** $f^{\wedge}=5.405$
Path Parasitic Delay: $P=8$ **Total path delay D is:** $D=29.620$

6 Stage Row Decoder: INV-INV-NAND3-INV-NAND3-INV



Path logical effort G: $G=25/9$ **Path Effort F:** $F=853.33$ **Optimal stage effort f^:** $f^{\wedge}=3.08$
Path Parasitic Delay: $P=10$ **Total path delay D is:** $D=28.479$

The following table summarizes our latest results for this question

Design (Functional Description)	Stages N	H (Constant) Path electrical Effort	B (Constant) Path branching Effort	G Path logical Effort	F Path effort	P Path parasitic Delay	$\hat{f}$ Stage Optimal Effort	D Path Delay
NAND5-INV	2	19.2	16	7/3	716.80	6	26.77	59.55
INV-NAND5-INV	3	19.2	16	7/3	716.80	7	8.95	33.85
INV-INV-NAND5-INV	4	19.2	16	7/3	716.80	8	5.17	28.70
NAND3-INV-NAND3-INV	4	19.2	16	25/9	853.33	8	5.40	29.62
INV-NAND3-INV-NAND3-INV	5	19.2	16	25/9	853.33	9	3.86	28.28
INV-INV-NAND3-INV-NAND3-INV	6	19.2	16	25/9	853.33	10	3.08	28.48

1.2 Use stick diagrams to estimate the area of each decoder in lambda and microns. Size the gates based on the corresponding logical effort for each gate.

note: given that each address line can drive (only) 10 units of capacitance and the fact that we have a branching of 16 at the intermediate node prior to the NAND gate, this means our address inputs would not optimally be able to drive NAND inputs unbuffered, for otherwise the address inputs would have to drive at least 16 units of capacitance (or said otherwise transistors sizes at inputs of NAND gates would have to be smaller than 1 unit). We have tried to compensate for this: in designs with odd number of stages, we brought the additional inverters to the inputs to aid in buffering in driving our NAND inputs; Notwithstanding all calculations are useful for comparing all of our other decoders with our optimal 5 stage decoder. (see summary table at the end of this section)

Starting in order from our 2 stage decoder for reference:

2 Stage Row Decoder: NAND5 X – INV Y

From our prior calculations we know our optimal stage effort $\hat{f}$: $\hat{f}=26.773$
Recall, reasonable stage efforts range from 2.4 to 6, hence we know this impractically high stage effort will results in an impractical non-optimal design.

Working backwards to find input capacitances/gate sizes, recall:

$$\hat{f}=gh=g\frac{C_{out}}{C_{in}} \quad \text{therefore} \quad C_{in\_y}=g_{inv}\frac{C_{out\_y}}{\hat{f}}=\frac{192}{26.77}=7.172 \quad \text{and}$$

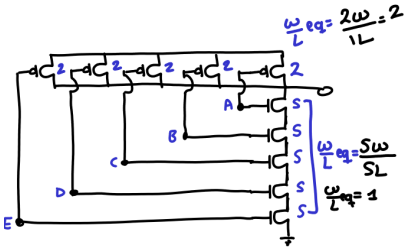
$$C_{in\_x}=g_{nand5}\frac{C_{out\_x}}{\hat{f}}=7/3\,x\,\frac{7.172}{26.77}=0.625$$

At this point, we continue working backward to find transistor sizes. For this purpose, we need to recall P:N ratios necessary for each gate, mainly:

- Inverter Y P/N: 2/1
- NAND5 X P/N: 2/5

As an example for the NAND5 gate P/N ratio: we know g=7/3 and by definition: *logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current* $g=\frac{C_{in\_gate}}{C_{in\_inverter}}$ In this case and given our NAND5 gate transistors schematic

The input unit capacitance looking into one terminal must be 7, additionally the equivalent W/L should be equivalent to that of an inverter 2/1 response, as outlined in the right.



Working back we can find the size of the P and N transistors in units of our unit transistor size.

INV Y: We know the input capacitance for Y (above) is 7.172 units, and we know the sizes must respect the P:N ratio of 2 to 1 for an inverter, therefore we can calculate: $W_p = \frac{7.172}{3} x\, 2 = 4.7813$

$W_n = \frac{7.172}{3} x\, 1 = 2.3907$ whereas Wp and Wn should add to INV y size as expected.

NAND5 X: We know the input capacitance for X = 0.625 and we know it must respect the P:N ratio of 2 to 5 for a NAND5, therefore we can calculate: $W_p = \frac{0.625}{7} x\, 2 = 0.1786$ $W_n = \frac{0.625}{7} x\, 5 = 0.4464$ Whereas Wp and Wn should add to NAND5 X size as expected.

2-Stage: NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
NAND5 X	0.625	0.179	0.446
INV Y	7.172	4.781	2.391

3 Stage Row Decoder: INV X – NAND5 Y - INV Z

Following the exact same reasoning as above for calculating our gate capacitances in units:

Given $f^{\wedge} = 8.95$, then $C_{in\_z} = g_{inv} \frac{C_{out\_z}}{f^{\wedge}} = \frac{192}{8.95} = 21.453$

$C_{in\_y} = g_{nand5} \frac{C_{out\_y}}{f^{\wedge}} = g_{nand5} \frac{C_{in\_z}}{f^{\wedge}} = 7/3\, x\, \frac{21.453}{8.95} = 5.593$

$C_{in\_x} = g_{inv} \frac{(C_{out\_x} b_x)}{f^{\wedge}} = g_{inv} \frac{(C_{in\_y} b_x)}{f^{\wedge}} = \frac{(5.593\, x\, 16)}{8.95} = 9.9987$ which should be close to ~10 as expected given that each address line can drive up to 10 units of capacitance. Subsequently, we find the gate N and P transistor sizes in units, working backwards:

INV Z: We know the input capacitance for Z (above) is 21.453 units, and we know the sizes must respect the P:N ratio of 2 to 1 for an inverter, therefore we can calculate: $W_p = \frac{z}{3} x\, 2 = 14.302$

$W_n = \frac{z}{3} x\, 1 = 7.151$ similarly,

NAND5 Y: We know the input capacitance for Y (above) is 5.593 units, and we know the sizes must respect the P:N ratio of 2 to 5 for a NAND5, therefore we can calculate: $W_p = \frac{y}{7} x\, 2 = 1.598$

$W_n = \frac{y}{7} x\, 5 = 3.995$

INV X: We know the input capacitance for X (above) is 9.999 units, and we know the sizes must respect the P:N ratio of 2 to 1 for an inverter, therefore we can calculate: $W_p = 6.666$ $W_n = 3.333$

3-Stage: INV-NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
INV X	9.999	6.666	3.333
NAND5 Y	5.593	1.598	3.995
INV Z	21.453	14.302	7.151

4 Stage Row Decoder (4a): INV W – INV X – NAND5 Y - INV Z

The process for finding gate input capacitance in units and subsequently the gate transistor sizes for all subsequent decoder designs employs the exact same steps (working backward from the optimal stage effort to find the gates input capacitances in units, then onwards from the gate capacitance size we can find the P and N transistors sizes respecting their appropriate ratio for each type of gate, as illustrated previously). therefore we decided to move to the use of a spreadsheet to calculate our results in a more organized fashion. All results for other decoder designs can be found below:

4a-Stage: INV-INV-NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
INV W	10.033	6.689	3.344
INV X	51.872	34.581	17.291
NAND5 Y	16.761	4.789	11.972
INV Z	37.137	24.758	12.379

4 Stage Row Decoder (4b): NAND3 W – INV X – NAND3 Y - INV Z

4b-Stage: NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
NAND3 W	0.627	0.251	0.376
INV X	2.032	1.355	0.677
NAND3 Y	10.974	4.390	6.584
INV Z	35.556	23.704	11.852

5 Stage Row Decoder: INV V – NAND3 W – INV X - NAND3 Y – INV Z

5-Stage: INV-NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
INV V	9.958	6.639	3.319
NAND3 W	2.402	0.961	1.441
INV X	5.564	3.709	1.855
NAND3 Y	21.477	8.591	12.886
INV Z	49.741	33.161	16.580

6 Stage Row Decoder: INV U – INV V – NAND3 W - INV X – NAND3 Y – INV Z

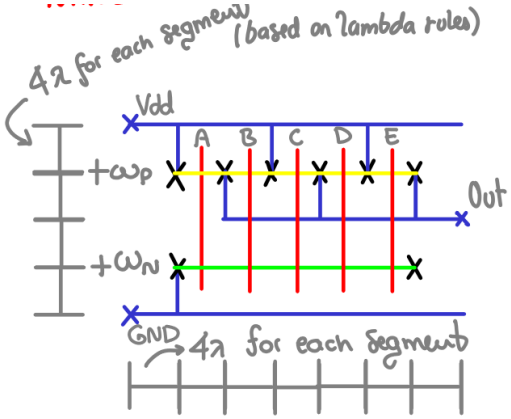
6-Stage: INV-INV-NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)
INV U	9.995	6.663	3.332
INV V	30.784	20.523	10.261
NAND3 W	5.926	2.370	3.556
INV X	10.952	7.301	3.651
NAND3 Y	33.733	13.493	20.240
INV Z	62.338	41.559	20.779

At this point, we needed to make use of stick diagrams to estimate approximate areas for each decoder design. (in lambda and microns) (recall Unit transistor of 4λ/2λ: λ = 0.2 µm)

The general approach we followed for this section was to start by calculating the area for each type of gate (with its respective sizes and following lambda rules), then multiple this gate area times the number of each types of gates for the complete decoder, i.e. for each gate:

NAND5: Expression for Area



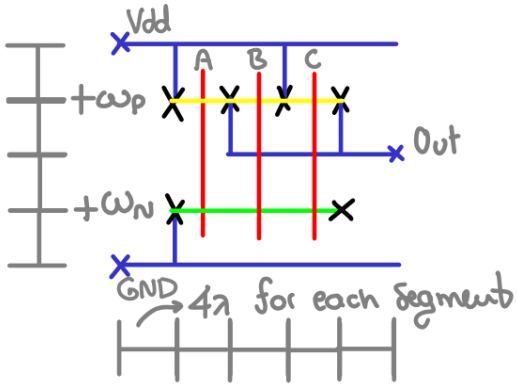
$$Y_{dim} = (4\,x\,4\,\lambda) + (W_{pn5}\,x\,4\,\lambda) + (W_{nn5}\,x\,4\,\lambda)$$
$$Y_{dim} = 4\,\lambda\,(4 + W_{pn5} + W_{nn5})$$

$$X_{dim} = 7\,x\,4\,\lambda = 28\,\lambda$$

Therefore we can calculate, area per NAND5 gate:

$$A_{NAND5} = X_{dim}\,Y_{dim} = 28\,\lambda\,x\,4\,\lambda\,(4 + W_{pn5} + W_{nn5})$$
$$A_{NAND5} = 112\,(4 + W_{pn5} + W_{nn5})\,\lambda^2$$

NAND3: Expression for Area



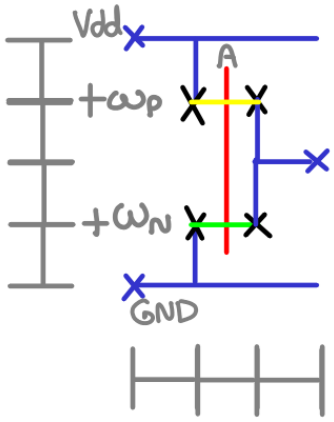
$$Y_{dim} = (4\,x\,4\,\lambda) + (W_{pn3}\,x\,4\,\lambda) + (W_{nn3}\,x\,4\,\lambda)$$
$$Y_{dim} = 4\,\lambda\,(4 + W_{pn3} + W_{nn3})$$

$$X_{dim} = 5\,x\,4\,\lambda = 20\,\lambda$$

Therefore we can calculate, area per NAND3 gate:

$$A_{NAND3} = X_{dim}\,Y_{dim} = 20\,\lambda\,x\,4\,\lambda\,(4 + W_{pn3} + W_{nn3})$$
$$A_{NAND3} = 80\,(4 + W_{pn3} + W_{nn3})\,\lambda^2$$

INV: Expression for Area



$$Y_{dim} = (4\,x\,4\,\lambda) + (W_{pinv}\,x\,4\,\lambda) + (W_{ninv}\,x\,4\,\lambda)$$
$$Y_{dim} = 4\,\lambda\,(4 + W_{pinv} + W_{ninv})$$

$$X_{dim} = 3\,x\,4\,\lambda = 12\,\lambda$$

Therefore we can calculate, area per NAND3 gate:

$$A_{INV} = X_{dim}\,Y_{dim} = 12\,\lambda\,x\,4\,\lambda\,(4 + W_{pinv} + W_{ninv})$$
$$A_{INV} = 48\,(4 + W_{pinv} + W_{ninv})\,\lambda^2$$

At this point, we have generic expressions for the area of each type of gate, hence knowing the number of gates of each type in a given decoder, we can calculate its total area, let's take for example our 3-stage decoder design for reference (helpful for illustration as it includes the line buffers):

3 Stage Row Decoder: INV X – NAND5 Y - INV Z

Per row-line we have:

- 1 INV Z
- 1 NAND5 Y Gate

And for all address lines we have:

- 10 INV X

And we have 32 row lines in our decoder, therefore the total number of INV Z instances and NAND5Y gates is 32 of each type respectively, we can tabulate our results as:

$$Total\ Area of\ Z\ Inverters = 32\ x\ A_{invz} \qquad Total\ Area of\ NAND\,5\ Y\ Gates = 32\ x\ A_{nand5\ y}$$
$$Total\ Area of\ X\ Inverters = 10\ x\ A_{invx}$$

And using our expressions for area per gate type (from stick diagrams above) together with our previously calculated required transistor sizes in units we can compute our total decoder area:

3-Stage: INV-NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
INV X	9.999	6.666	3.333	671.9376	26.877504	10	6719.376	268.775
NAND5 Y	5.593	1.598	3.995	1074.416	42.97664	32	34381.312	1375.252
INV Z	21.453	14.302	7.151	1221.744	48.86976	32	39095.808	1563.832
						Full Decoder	80196.496	3207.860

The process to calculate the decoder areas for all other designs, is repeated as above, being mindful of the type of gate (and respective expression to be used) as well as the P and N transistor sizes in units; we employed our spreadsheet to tabulate our values and avoid mistakes.

2 Stage Row Decoder: NAND5 X – INV Y

2-Stage: NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
NAND5 X	0.625	0.179	0.446	518	20.72	32	16576	663.04
INV Y	7.172	4.781	2.391	536.256	21.450	32	17160.192	686.408
						Full Decoder	33736.192	1349.448

4 Stage Row Decoder (4a): INV W – INV X – NAND5 Y - INV Z

4a-Stage: INV-INV-NAND5-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
INV W	10.033	6.689	3.344	673.584	26.94336	10	6735.840	269.434
INV X	51.872	34.581	17.291	2681.856	107.27424	10	26818.560	1072.742
NAND5 Y	16.761	4.789	11.972	2325.232	93.00928	32	74407.424	2976.297
INV Z	37.137	24.758	12.379	1974.576	78.98304	32	63186.432	2527.457
						Full Decoder	171148.256	6845.930

4 Stage Row Decoder (4b): NAND3 W – INV X – NAND3 Y - INV Z

4b-Stage: NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
NAND3 W	0.627	0.251	0.376	370.160	14.8064	32	11845.120	473.805
INV X	2.032	1.355	0.677	289.536	11.58144	32	9265.152	370.606
NAND3 Y	10.974	4.390	6.584	1197.920	47.9168	32	38333.440	1533.338
INV Z	35.556	23.704	11.852	1898.688	75.94752	32	60758.016	2430.321
						Full Decoder	120201.728	4808.069

5 Stage Row Decoder: INV V – NAND3 W – INV X - NAND3 Y – INV Z

5-Stage: INV-NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
INV V	9.958	6.639	3.319	669.984	26.79936	10	6699.840	267.994
NAND3 W	2.402	0.961	1.441	512.160	20.4864	32	16389.120	655.565
INV X	5.564	3.709	1.855	459.072	18.36288	32	14690.304	587.612
NAND3 Y	21.477	8.591	12.886	2038.160	81.5264	32	65221.120	2608.845
INV Z	49.741	33.161	16.580	2579.568	103.18272	32	82546.176	3301.847
						Full Decoder	185546.560	7421.862

6 Stage Row Decoder: INV U – INV V – NAND3 W - INV X – NAND3 Y – INV Z

6-Stage: INV-INV-NAND3-INV-NAND3-INV

Gate	In Capacitance (unit)	Wp (unit)	Wn (unit)	Area per Gate (λ^2)	Area per Gate (um^2)	# of Gates	Total Area (λ^2)	Total Area (um^2)
INV U	9.995	6.663	3.332	671.760	26.8704	10	6717.600	268.704
INV V	30.784	20.523	10.261	1669.632	66.78528	10	16696.320	667.853
NAND3 W	5.926	2.370	3.556	794.080	31.7632	32	25410.560	1016.422
INV X	10.952	7.301	3.651	717.696	28.70784	32	22966.272	918.651
NAND3 Y	33.733	13.493	20.240	3018.640	120.7456	32	96596.480	3863.859
INV Z	62.338	41.559	20.779	3184.224	127.36896	32	101895.168	4075.807
						Full Decoder	270282.400	10811.296

1.3 Calculate (average) dynamic power dissipation for each design assuming “rolling” address incrementing monotonically from all zeros to all ones in 32 usec.

If we assume that every address from 0 to 31 is equally probable, over every 32 uS period, we would have 32 random addresses selection (i.e. 32 random row selections). Now assuming a rolling address selection, we can infer that each row will be selected once every 32 uS. Hence we can calculate the average switching clock of the gates in each row decoder and address line, then knowing both the gate effective switching frequency and the node capacitance it has to drive, we can calculate the specific

gate switching power. Following a similar procedure as before then, we can tabulate the switching power per gate for all of the gates in the decoder (mindful of the specific switching power consumption for each gate type) and arrive at an estimate for the approximate switching power of the decoder design.
Canonically, the switching power per gate can be expressed as: $P_{switching} = Vdd^2 C f_{sw}$
Note that here the supply voltage (Vdd=3.3V), Fsw is the effective switching frequency of the gate, and C is the capacitive load the gate has to drive.

Now that compared to the other expression for switching power $P_{switching} = Vdd^2 C \alpha f_{clock}$, αf_{clock} actually calculating the effective switching frequency Fsw above, this expression is more useful for larger systems where gate-wise accounting of power is impractical and instead an approximate value for α for a given system clock is known.

Each gate has to drive the gate of the transistors of the gate(s) that follows it (it has to charge and discharge the load capacitance), in our case we can come up with an expression for the gate capacitance (in fF per unit) assuming a 350nm process (data from table 8.5, Wunit = 4λ), where:

$C_{gate} = Width_{total}\, x\, (C_{\text{parasitic\_micron}})$ $C_{gate} = (C_{units}\, x\, 4\,\lambda)(0.2\, um/\lambda)(C_{\text{g(power)}} + C_{\text{d(isolated)}})$
$C_{gate} = C_{units}\, 0.8\, um\, x\, (2.20\, fF/um + 1.63\, fF/um) = C_{units}\, x\, 0.8\, um\, x\, 3.83\, fF/um$
Therefore we have our expression for gate capacitance per um:
$C_{gate} = C_{units}\, x\, 3.064\, fF$

Therefore our expression for switching power per gate becomes:
$P_{\text{switching\_gate}} = C_{\text{units\_gate\_load}}\, x\, 3.064\, fF\, x\, f_{\text{sw\_gate}}\, x\, 3.3\, V^2$

i.e. intuitively, if we know the switching output frequency of each gate (how often it switches logic levels) and we know the total units of capacitance it drives (the load) then we can tabulate with an spreadsheet the switching power per gate for each gate type (with it's own switching and output load properties), then calculate times the total number of gates of each type in the decoder and aggregate our results to calculate total switching power. Additionally assuming a ~10% extra for short circuit power, we can calculate dynamic power as: $P_{\text{dynamic}} \simeq P_{switching} + 0.10\, P_{switching}$

Now to use our formula for switching power per gate, we have the capacitance in units from before, but we need to calculate the effective switching frequency of each type of gate.

3 Stage Row Decoder: INV X – NAND5 Y - INV Z

Using again our 3-stage row decoder as an example to illustrate our process (which we integrated in our spreadsheet for calculations). First, we need to figure out the equivalent switching frequency of each gate along the row-path and for the line buffers, in our case:
$f_{\text{sw\_out\_z}} = f_{\text{sw\_out\_y}} = 31.25\, KHz$ for the row gates (once every 32 uS), and for each of the adx line buffers (true and complementary) starting from LSB $f_{\text{sw\_out\_x0}} = 16\, x\, (31.250\, KHz) = 500\, KHz$
$f_{\text{sw\_out\_x1}} = 8\, x\, (31.250\, KHz) = 250\, KHz$ $f_{\text{sw\_out\_x2}} = 4\, x\, (31.250\, KHz) = 125\, KHz$
$f_{\text{sw\_out\_x3}} = 2\, x\, (31.250\, KHz) = 62.5\, KHz$ $f_{\text{sw\_out\_x4}} = 31.250\, KHz$

In fact, we can use the average switching frequency over all X Inverters (which was proven to be equivalent) as our average switching frequency for power calculations: $f_{\text{sw\_out\_xavg}} = 193.75\, KHz$

Having the effective switching frequency of each gate, we can calculate our switching power consumption and for all gates as:

- Note out load capacitance for each gate is that of the next stage, as discussed above, it is important to be aware of branching as well
- Note the out/load capacitance is then multiplied by 3.064fF to convert to fF as explained above
- The total switching power then is simply the switching power per gate type times number of gates
- Total dynamic power is simply 10% extra to account for short circuit power

Power Calculation						
Gate	Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power	Total Dynamic Power
INV X	89.488	274.191	193750	5.785E-07	5.785E-06	6.364E-06
NAND5 Y	21.453	65.732	31250	2.237E-08	7.158E-07	7.874E-07
INV Z	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	1.291E-05	1.420E-05

2 Stage Row Decoder: NAND5 X – INV Y

Where Cg = 2.20fF/um, Cd = 1.63fF/um, Vdd = 3.3V

Power Calculation						
Gate	Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power	Total Dynamic Power
NAND5 X	7.172	21.975	31250	7.478E-09	2.393E-07	2.632E-07
INV Y	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	6.646E-06	7.310E-06

4 Stage Row Decoder (4a): INV W – INV X – NAND5 Y - INV Z

Power Calculation						
Gate	Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power	Total Dynamic Power
INV W	51.872	158.936	193750	3.353E-07	3.353E-06	3.689E-06
INV X	268.176	821.691	193750	1.734E-06	1.734E-05	1.907E-05
NAND5 Y	37.137	113.788	31250	3.872E-08	1.239E-06	1.363E-06
INV Z	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	2.834E-05	3.117E-05

4 Stage Row Decoder (4b): NAND3 W – INV X – NAND3 Y - INV Z

Power Calculation						
Gate	Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power (W)	Total Dynamic Power (W)
NAND3 W	2.032	6.226	125000	8.475E-09	2.712E-07	2.983E-07
INV X	10.974	33.624	125000	4.577E-08	1.465E-06	1.611E-06
NAND3 Y	35.556	108.944	31250	3.707E-08	1.186E-06	1.305E-06
INV Z	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	9.329E-06	1.026E-05

5 Stage Row Decoder: INV V – NAND3 W – INV X - NAND3 Y – INV Z

Gate	Power Calculation Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power (W)	Total Dynamic Power (W)
INV V	38.432	117.756	193750	2.485E-07	2.485E-06	2.733E-06
NAND3 W	5.564	17.048	125000	2.321E-08	7.426E-07	8.169E-07
INV X	21.477	65.806	125000	8.958E-08	2.866E-06	3.153E-06
NAND3 Y	49.741	152.406	31250	5.187E-08	1.660E-06	1.826E-06
INV Z	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	1.416E-05	1.558E-05

6 Stage Row Decoder: INV U – INV V – NAND3 W - INV X – NAND3 Y – INV Z

Gate	Power Calculation Out/load Capacitance (units)	Out/load Capacitance* (fF)	Fsw Out Switch Frequency	Power Switching	Total Switching Power (W)	Total Dynamic Power (W)
INV U	30.784	94.322	193750	1.990E-07	1.990E-06	2.189E-06
INV V	94.816	290.516	193750	6.130E-07	6.130E-06	6.743E-06
NAND3 W	10.952	33.557	125000	4.568E-08	1.462E-06	1.608E-06
INV X	33.733	103.358	125000	1.407E-07	4.502E-06	4.952E-06
NAND3 Y	62.338	191.004	31250	6.500E-08	2.080E-06	2.288E-06
INV Z	192	588.288	31250	2.002E-07	6.406E-06	7.047E-06
				Full Decoder	2.257E-05	2.483E-05

1.4 Generate a comparison table for the 6 decoder designs

The following is our summary table summarizing our results for each decoder design, note as requested: the **smallest delay in bold**, the *smallest power in italics* and the <u>smallest areas underlined</u>.

Design (Functional Description)	G	P	D	Aλ²	Aμ²	$P_{dynamic}$(W)	PD
	Path logical Effort	Path parasitic Delay	Path Delay	Area in λ squared	Area in μm squared	Dynamic Power (watts)	Power delay Product
NAND5-INV	7/3	6	59.546	33,736	1349	*7.310E-06*	4.35E-04
INV-NAND5-INV	7/3	7	33.849	80,196	3208	1.420E-05	4.81E-04
INV-INV-NAND5-INV	7/3	8	28.697	171,148	6846	3.117E-05	8.94E-04
NAND3-INV-NAND3-INV	25/9	8	29.619	120,202	4808	1.026E-05	3.04E-04
INV-NAND3-INV-NAND3-INV	25/9	9	**28.284**	185,547	7422	1.558E-05	4.41E-04
INV-INV-NAND3-INV-NAND3-INV	25/9	10	28.479	270,282	10811	2.483E-05	7.07E-04

2. Schematic entry and transistor level simulation

2.1 Design Cadence Composer schematic and (self-describing) symbol views for inverters and NAND gates based on the transistor sizing obtained in question 1 above for the fastest decoder.

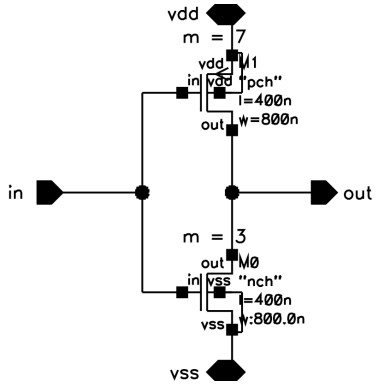
The fastest decoder as expected from our optimal number of stages estimation at the beginning of section 1 is our 5-stage decoder. Now in order to build our decoder gates at the transistor level, it is often useful to choose transistor sizes that are integer multiples of our unit sized transistor. (this makes simulation simpler as one can use multiple transistors in parallel m, or multiple fingers)

For our case, we will use our unit transistor of Wunit = 4λ/2λ, where λ = 0.2 µm. This means we need to approximate and round-off our current ideal transistor sizes to integer multiples of our unit transistor, in a way that is convenient to design with, We will use the following rounded integer values: (we tried as best to keep ratios constant where possible)

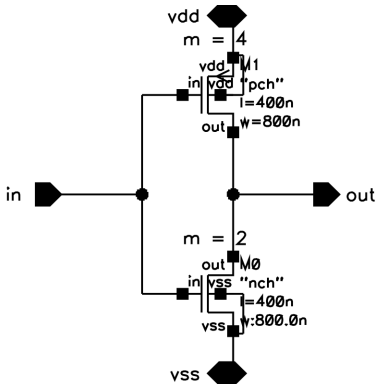
	Wp (units)	Lp (units)	Wp (λ)	Lp(λ)	Wp (um)	Lp (um)	Wn (units)	Ln (units)	Wn (λ)	Ln(λ)	Wn (um)	Ln (um)
INVV	7	1	28	2	5.6	0.4	3	1	12	2	2.4	0.4
NAND3W	1	1	4	2	0.8	0.4	1	1	4	2	0.8	0.4
INVX	4	1	16	2	3.2	0.4	2	1	8	2	1.6	0.4
NAND3Y	9	1	36	2	7.2	0.4	13	1	52	2	10.4	0.4
INVZ	33	1	132	2	26.4	0.4	17	1	68	2	13.6	0.4

The following is the transistor schematic of every gate cell:

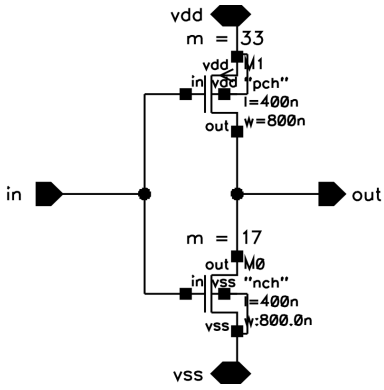
INV V Cell



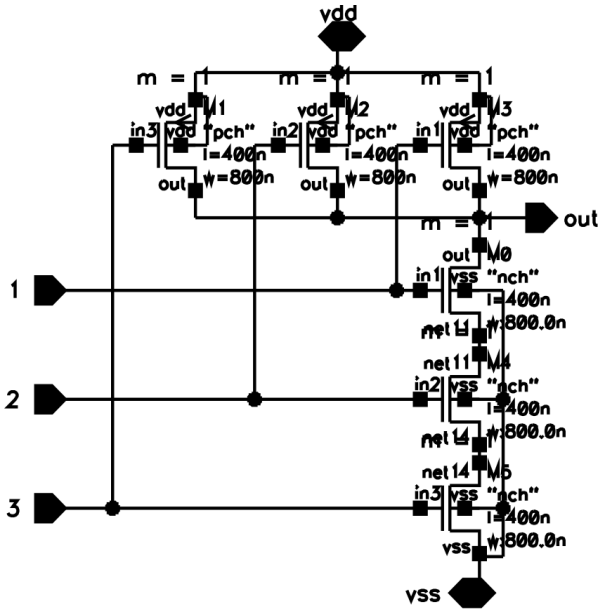
INV X Cell

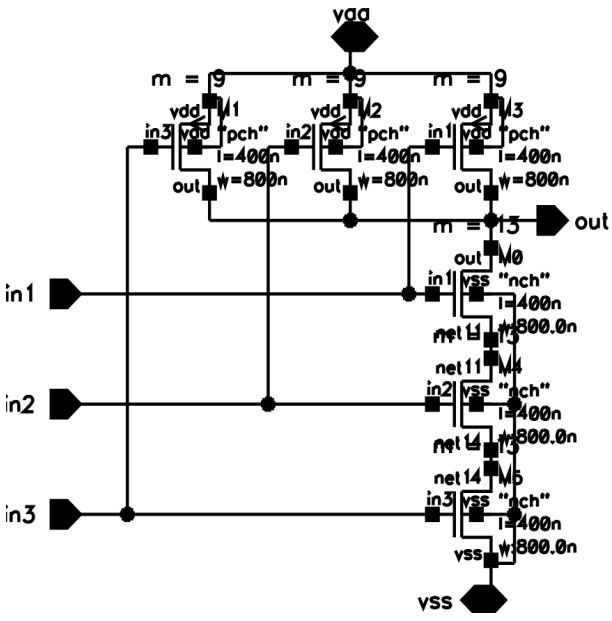


INV Z Cell

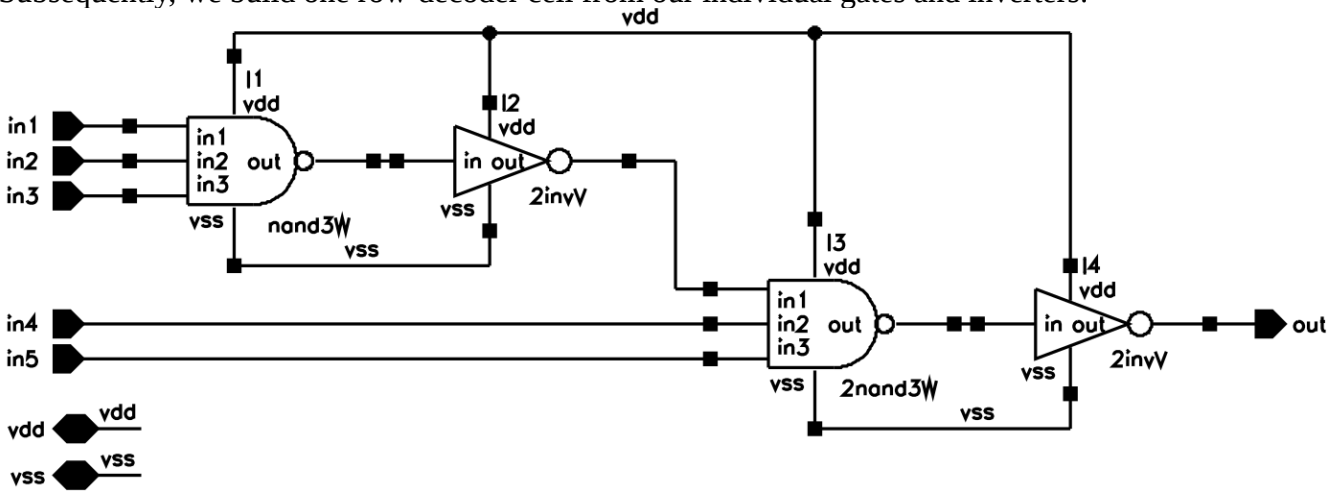


NAND3 W, NAND 3 Y (on the right)

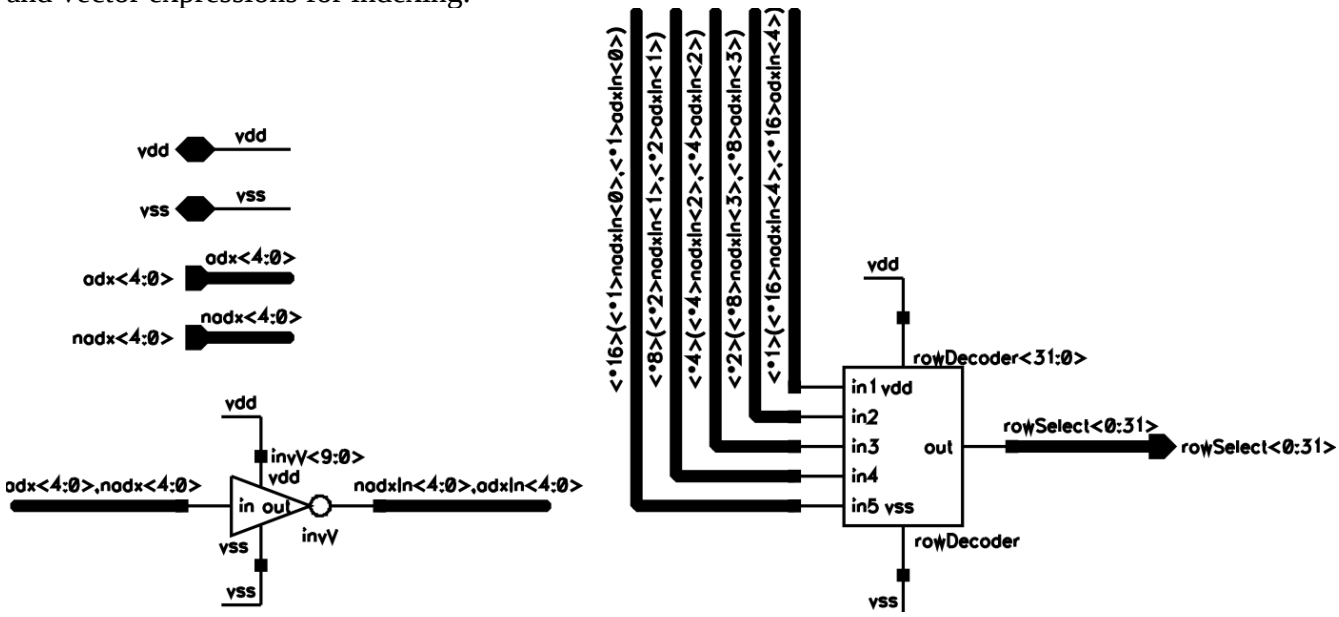




Subsequently, we build one row-decoder cell from our individual gates and inverters:



And we build our full decoder modularly and efficiently as an array of row decoder cells utilizing buses and vector expressions for indexing.

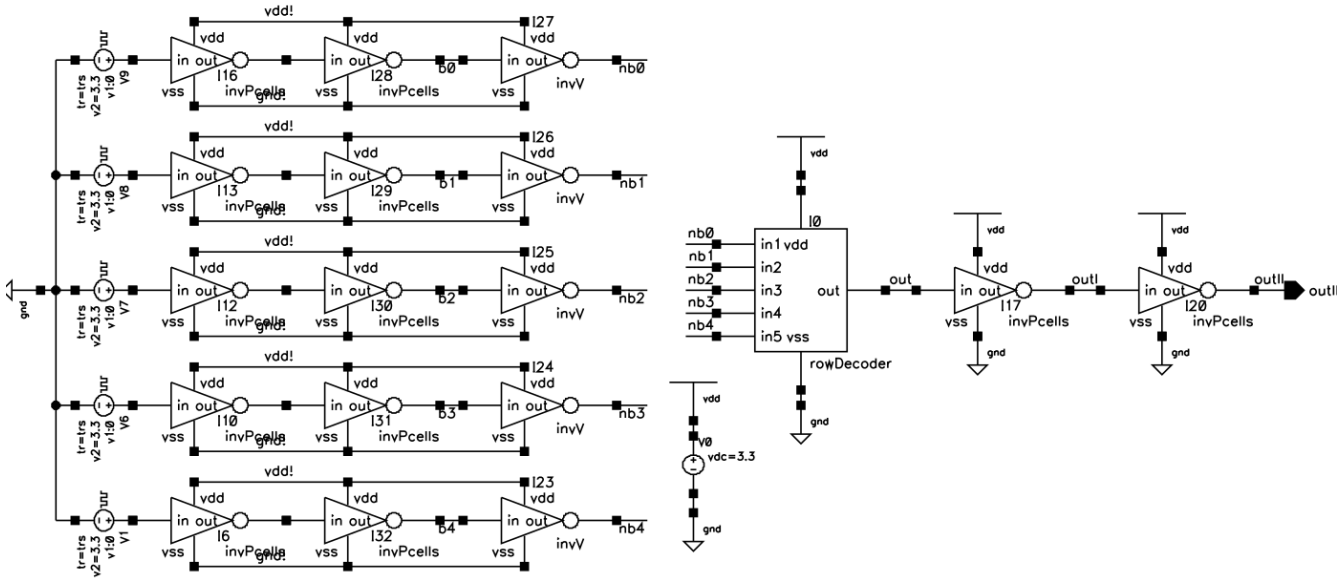


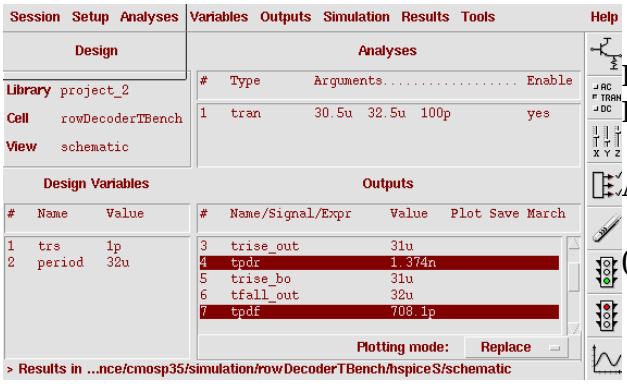
2.2 Validate the circuit delay value obtained in question 1 through simulation. Explain any observed differences.

Initially, to validate our propagation path delay, we built a testbench around a single row decoder cell and measured the path delay at the end of our digital code (0b’5 11111) from when a[0] (our LSB) switches high leading to our row selection output going active. (This can be more clearly seen from our testbench, the calculated result expression and transient simulations on next page)

Note we have both pre-shaping inverters as well as our expected load of 192 units (P:N 128:64) and load on load preceding it -- See testbench below.

Row decoder cell testbench





Propagation Rising Delay = 1.374nS
Propagation Falling Delay = 708.1pS

Average Propagation Path delay = 1.041nS

(compared to 1.1314nS, see below)

Now our estimated path delay was D = 28.284 in τ units. For a 350nm process (using reference table 8.8 in textbook) τ = 40pS, therefore our previously estimated path delay in amounts to:

$$D = 28.284\,x\,40\;pS = 1.1314\,nS$$

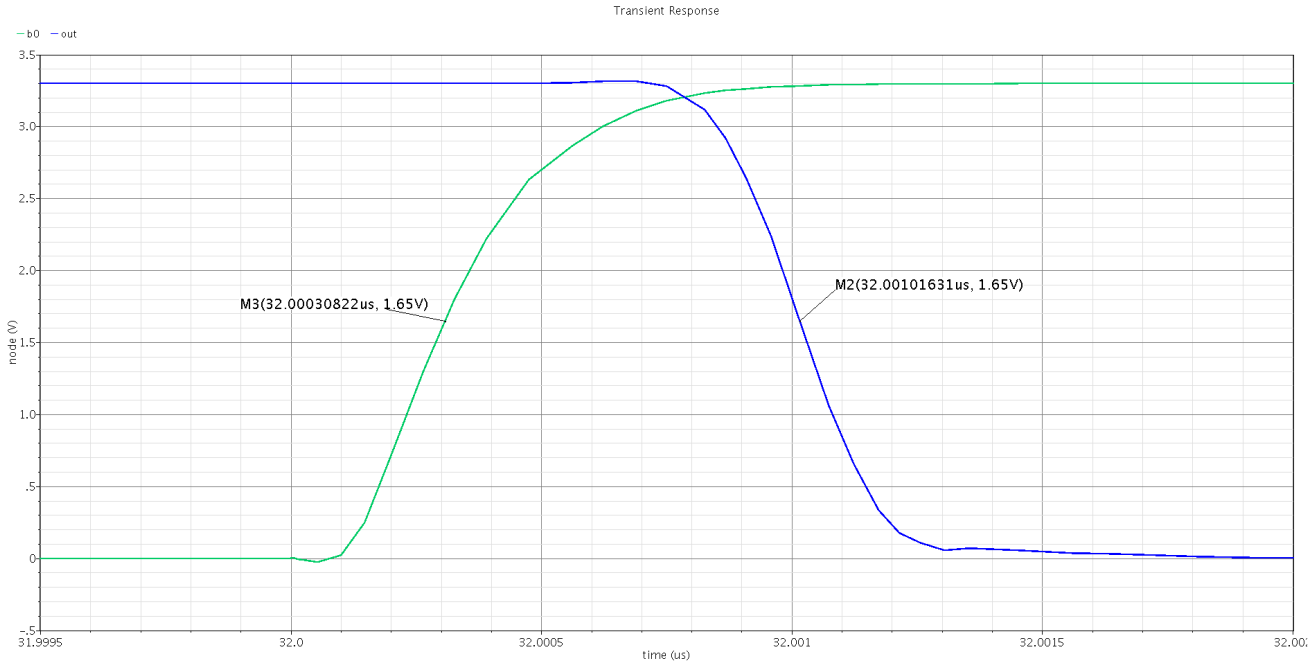
The following are our compared results from our recorded values (organized in spreadsheet format)

Row Decoder Path	Calculated D (τ)	τ (S)	Calculated D (S)	Measured tpdr	Measured tpdf	Measured Path Delay	Path Delay Error (%)
	Path Delay	τ for tsmc 350nm	Path Delay in S	Path propagation rising delay	Path propagation falling delay	Average rise and fall Propagation delay	error from Calculation to measurement
INVV-NAND3W-INVX-NAND3Y-INVZ	28.284	4E-11	1.13E-09	1.374E-09	7.081E-10	1.04E-09	8.67

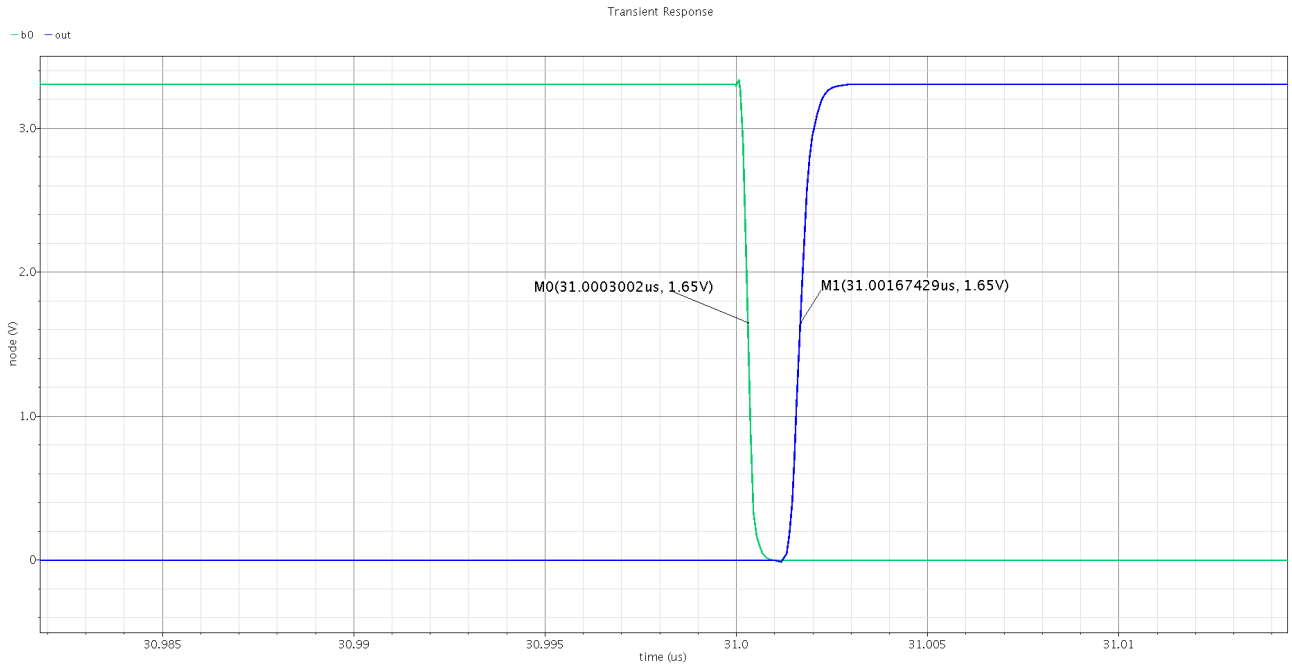
This value is very close to our measured delay, there is an error difference of 8.67% as compared to our measured delay.

- The small error could be due to integer-multiples of unit sizes used for our gate transistors (they were approximated to their closest integer)
- Another minor difference could arise due to the τ value approximation used for our 350nm process, which could vary slightly for our specific transistors.

Propagation Falling Delay

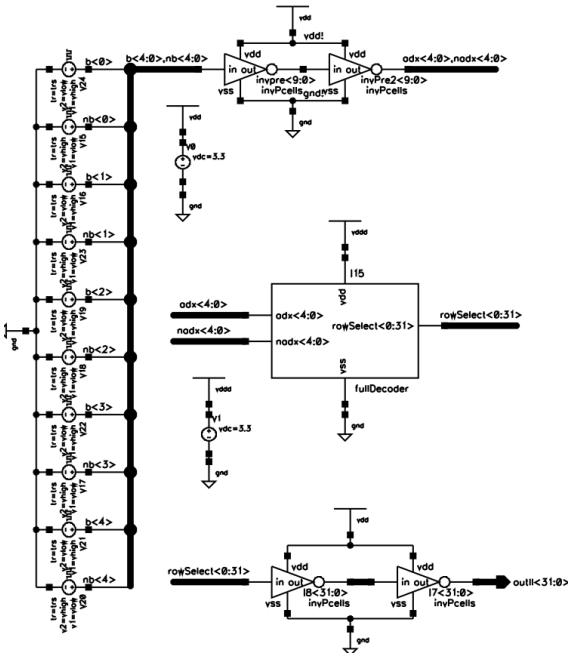


Propagation Rising Delay



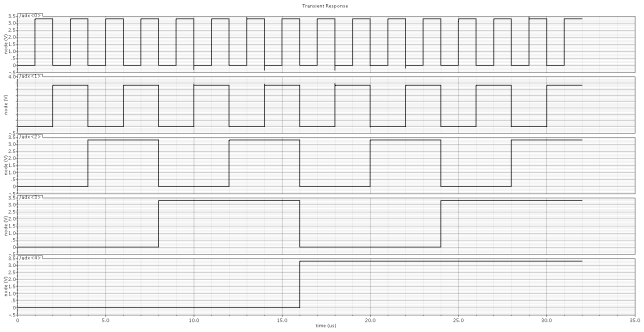
At this point we build a testbench around our full decoder to validate it's full functionality and assess it's power consumption:

Full Decoder Testbench

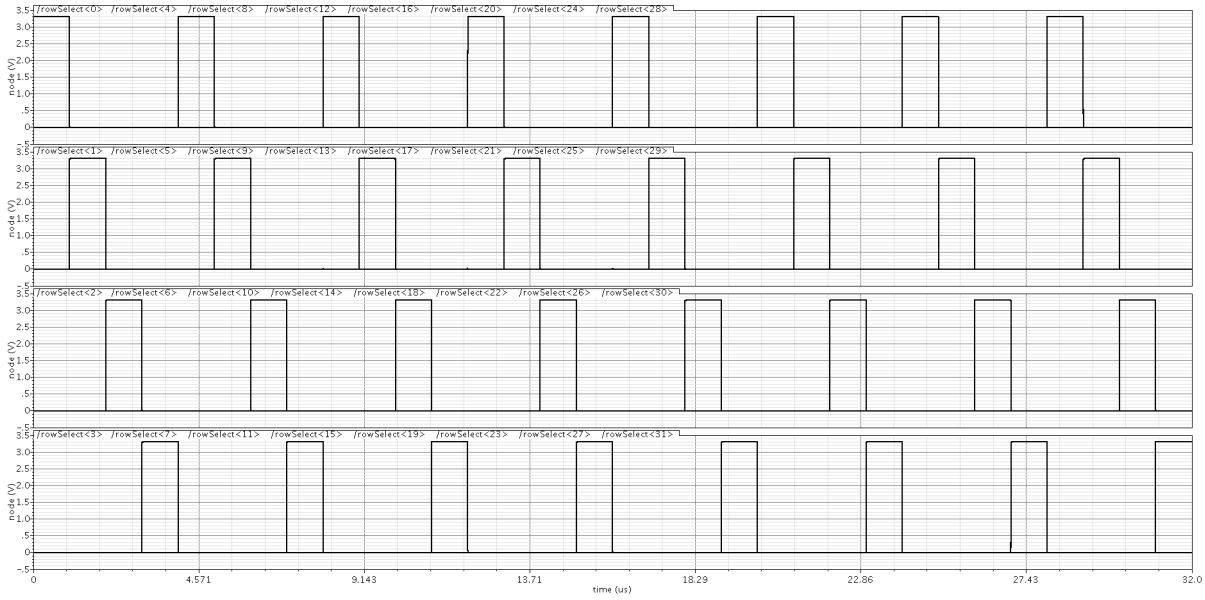


The full decoder testing results under a rolling address input can be seen below:

Rolling Address Input Digital Codes (standard test input)

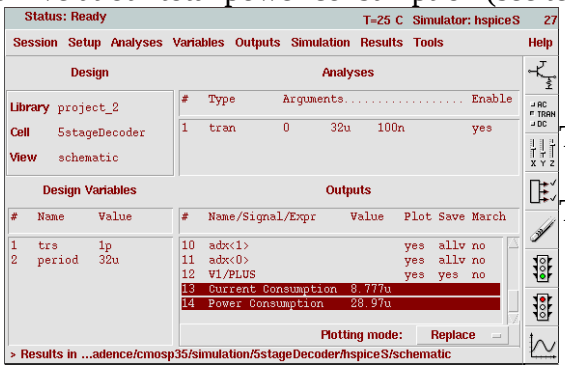


Full Decoder Row-selection Outputs (Overlaid on a single plot)



2.3 Obtain the simulated value for the total power dissipation of the circuit in 2.1 averaged across all combinations

For accurate current measurement purposes for our device under test, we decided to use a different 3.3V power supply to be used only for our full decoder (such that we can measure the current consumption of the decoder alone precisely by probing the dedicated power source). The transient current consumption is averaged over all addresses and multiplied times our supply voltage (3.3V) to arrive at our total power consumption (see testbench results below)



Total current consumption = 8.777uA

Total Power Consumption = 28.99 uW

2.4 Obtain the simulated value for the static power dissipation of the circuit in 2.1 averaged across all combinations of the address.

We can inspect average power consumption over every address – holding each address constant and then taking a measurement. Our static current consumption and power (due to leakage sources) is

Static Current Consumption = 10.93nA
Static Power Consumption = 36.1nW

Note these measurements were verified equivalently by measuring the power consumption at DC.

2.5 Validate the circuit dynamic power dissipation value obtained in question 1, by subtracting the result of question 2.4 from the result of question 2.3

At this point we recorded all of our power measurements and computed the dynamic power dissipations for comparison with our estimated values. We organized our results in a spreadsheet displayed below:

Power consumption	Total Current Consumption	Total Power Consumption	Static Current Consumption	Static Power Consumption	Dynamic Power Consumption
INVV-NAND3W-INVX-NAND3Y-INVZ	8.88E-06	2.93E-05	1.09E-08	3.61E-08	2.93E-05

Simulated Dynamic Power Consumption = 29.3uW

Note our **calculated dynamic power consumption of 15.58uW** versus our **simulated dynamic power of 29.3uW** are within the same order of magnitude (tens of uW) and are surprisingly close given that our estimations were calculated by hand using first-order approximations and in spite of our changes to make transistor sizing integer multiples of our unit size transistors – this could be the primary reason for the discrepancy of ~14uW.

3. Layout Design and Post-layout Verification

3.1 Layout the fastest decoder and perform DRC and LVS verification. Minimize the layout area while following standard cell layout style, and using proper transistor fingering techniques. Keep the design hierarchical and modular.

The general approach followed in our case for the layout of each cell – from the simplest to most complex – building up toward the row-decoder layout and subsequently towards the full decoder was as follows. (this process took a couple of iterations to get correctly, for instance initially all cells had a different cell height which made it difficult to fit compactly when putting together the row-decoder cell layout, so we needed to go back and modify all cells to adhere to a standard height)

The general process started as follows, for each cell:

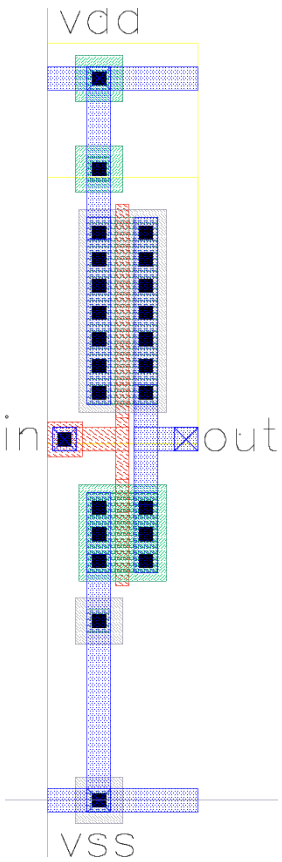
- Use stick diagrams Y dimension expression to figure out highest cell
  - This set our standard cell height approximately
  - This determine our largest cells in terms of area
- Use stick diagrams to guide our parallel (m) number of devices and fingering (if needed for large devices) strategy to optimize the use of area (compact cells)
  - Modify slightly device sizes if needed to allow for efficient multiplier and fingering combinations
- Enforce good grid usage to avoid off-grid errors, for this 350nm process technology
  - Min grid: 0.175 (technology λ)
  - Max grid: 0.875
  - Snap: 0.025 (min placement grid by technology DRC rules)
- Copy previous cell and flatten hierarchy to use as a starting reference
- Layout cell according to needed dimensions and multiplier/fingering strategy above
- DRC clean test
- Extraction
- ERC clean test
- LVS clean test
- For post-layout simulations (applicable test performed for row-decoder extracted cell and subsequently for full decoder extracted cell)
  - Generate pin-only schematic from extracted layout
  - Create respective symbol for pin-only schematic
  - Instantiate extracted cell symbol on testbench schematic
  - Modify simulation settings to allow simulation of extracted view with HSPICE

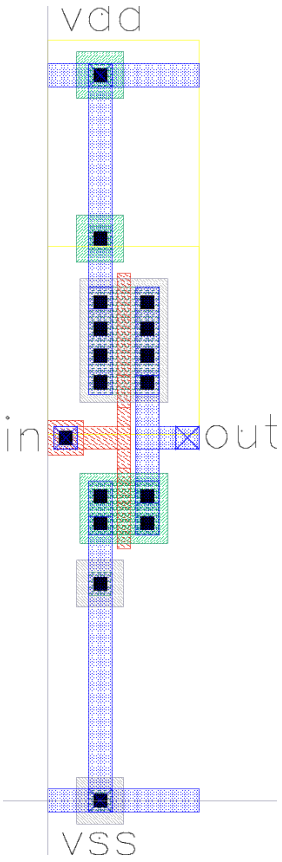
We settled on the following final device sizes to allow for efficient multiplier/fingering layout of our cells (red fields are those modified from simulations)

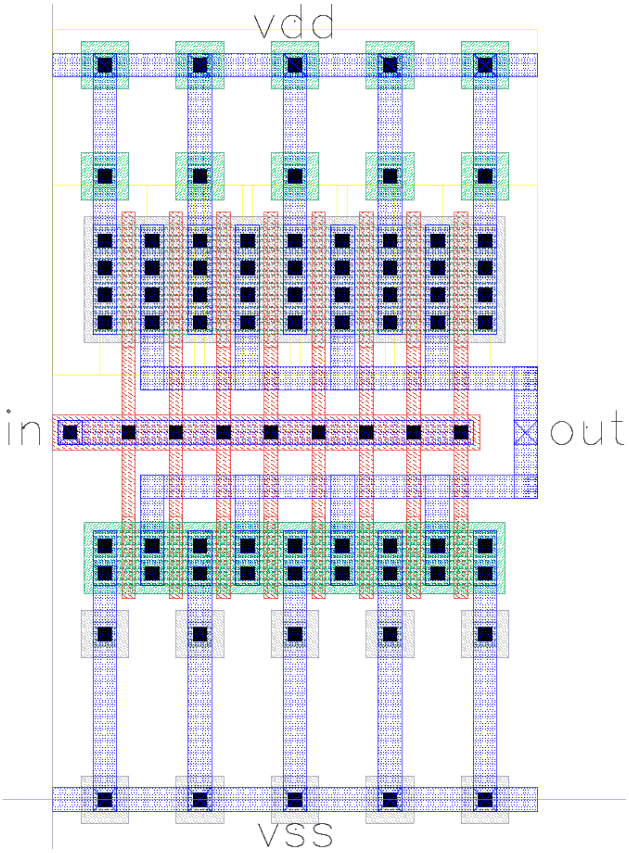
	Wp (units)	Lp (units)	Wp (λ)	Lp(λ)	Wp (um)	Lp (um)	Wn (units)	Ln (units)	Wn (λ)	Ln(λ)	Wn (um)	Ln (um)
INVV	7	1	28	2	5.6	0.4	3	1	12	2	2.4	0.4
NAND3W	1	1	4	2	0.8	0.4	1	1	4	2	0.8	0.4
INVX	4	1	16	2	3.2	0.4	2	1	8	2	1.6	0.4
NAND3Y	8	1	32	2	6.4	0.4	12	1	48	2	9.6	0.4
INVZ	32	1	128	2	25.6	0.4	16	1	64	2	12.8	0.4

Our standard cells height is 22.350um for all of them.

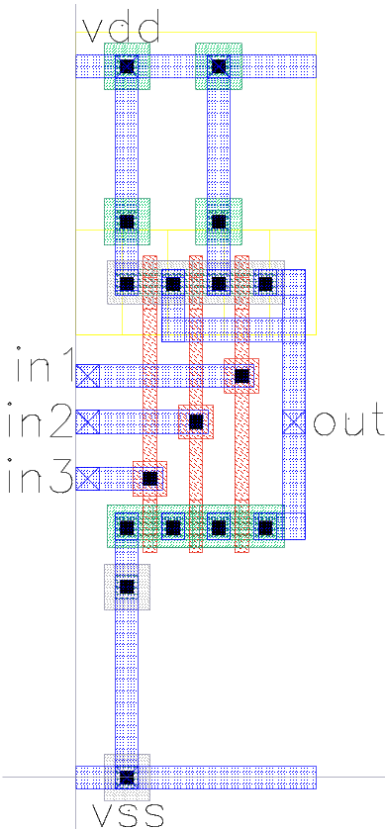
INV V Layout(Mp=7, Mn=3) **INV X Layout**(Mp=4, Mn=2) **INV Z Layout**(Mp=4, Nfp=8, Mn=2, Nfn=8)



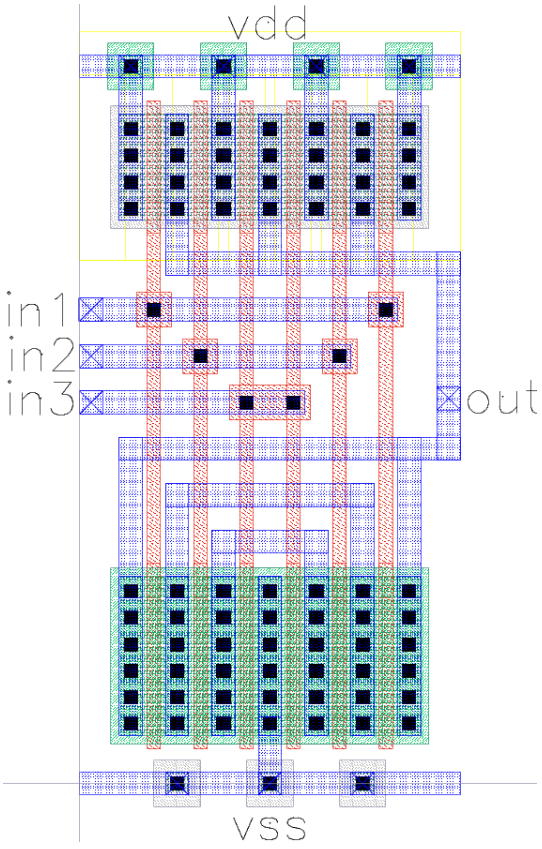




NAND3 W Layout(Mp=1,Mn=1)



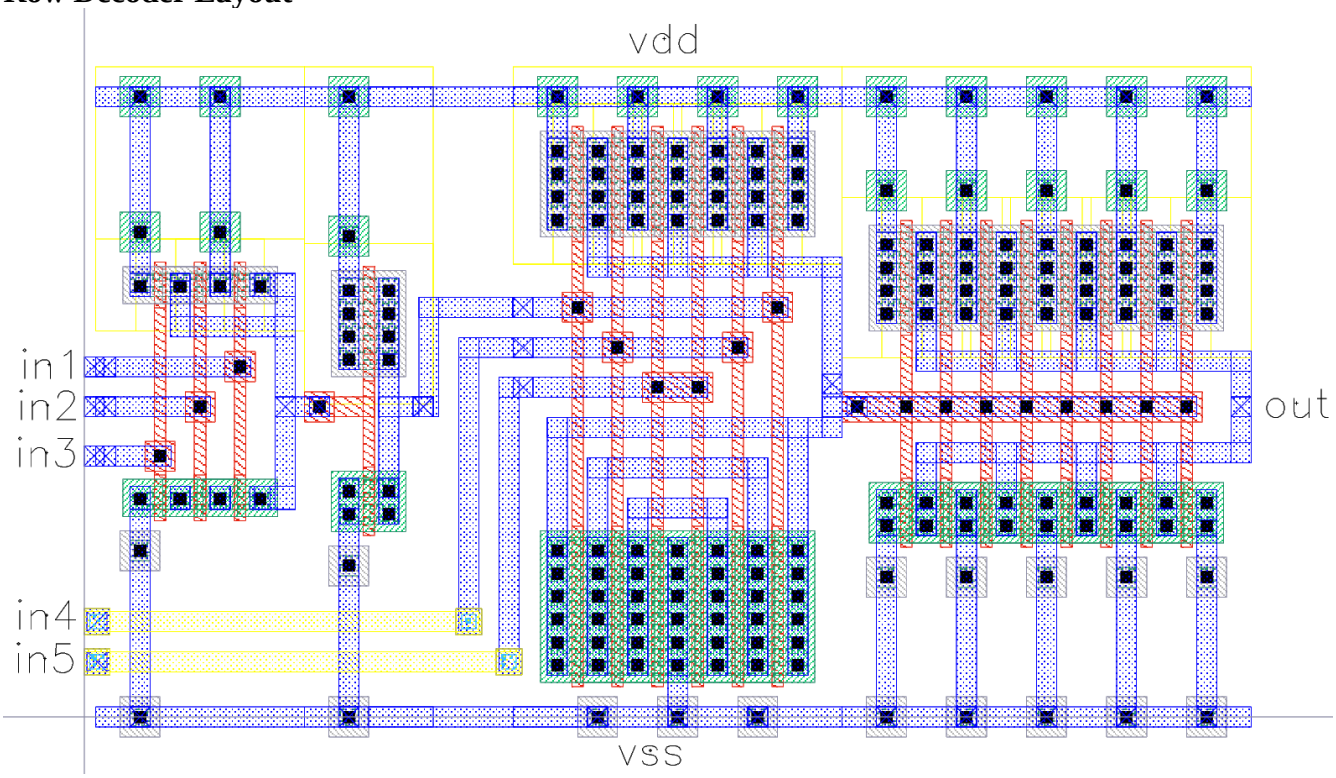
NAND3 Y Layout(Mp=4, Nfp=2, Mn=6, Nfn=2)



Note the fingering strategy for the NAND3Y gate is not standard, this took me sometime to understand, the symmetrical fingering pattern for the NMOS transistors must start from the center (where Vss is connected to at the bottom), this will become the shared source of our bottom-most NMOS transistor in the series chain in the NAND3 schematic, from this middle point in the layout it spreads sideward (and up the series chain in the NAND3 schematic) until it reaches the NMOS transistor closest to the output.

After having done the layout for each of the gates and inverters (and passed DRC, ERC and LVS), we use each of the layout cells to put together the layout of the row decoder as below:

Row Decoder Layout



Row Decoder Reports (DRC, ERC, LVS)

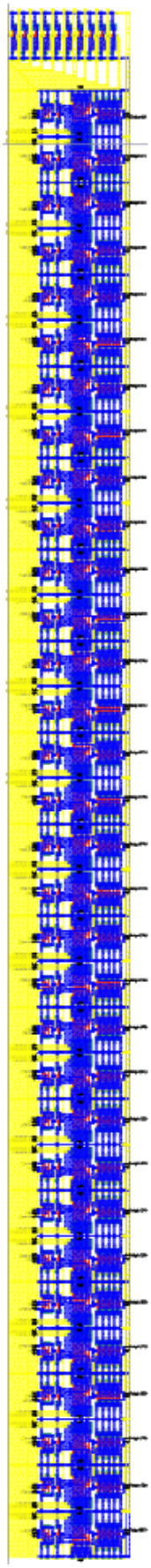
```
*********   Summary of rule violations for cell "rowDecoderLayout layout"   *********
   Total errors found: 0
```

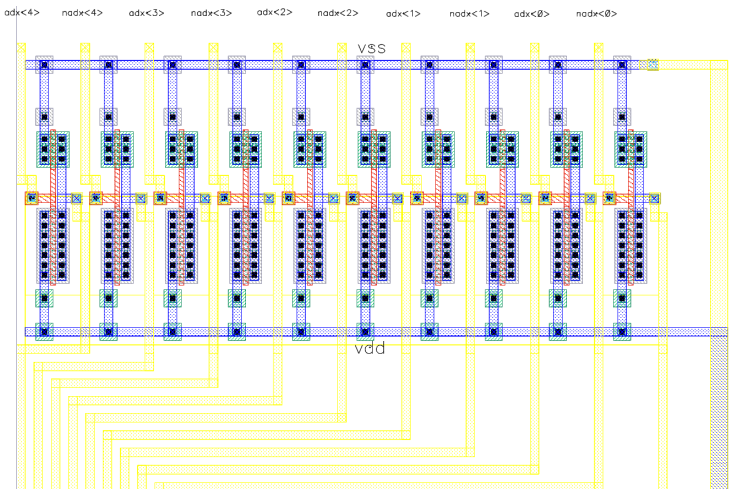
```
/ERC' that was started at 'Oct 24 16:23:12 2018' has succeeded
```

```
Begin netlist:    Oct 24 16:16:43 2018
    view name list  = ("auLvs" "extracted" "schematic")
    stop name list  = ("auLvs")
    library name    = "project_2"
    cell name   = "rowDecoderLayout"
    view name   = "extracted"
    globals lib = "basic"
Running Artist Flat Netlisting ...
End netlist:    Oct 24 16:16:43 2018

Moving original netlist to extNetlist
Removing parasitic components from netlist
    presistors removed:  0
    pcapacitors removed: 0
    pinductors removed:  0
    pdiodes removed:     0
    trans lines removed: 0
    15 nodes merged into 15 nodes


Begin netlist:    Oct 24 16:16:43 2018
    view name list  = ("auLvs" "schematic")
    stop name list  = ("auLvs")
    library name    = "project_2"
    cell name   = "rowDecoder"
    view name   = "schematic"
    globals lib = "basic"
Running Artist Flat Netlisting ...
End netlist:    Oct 24 16:16:43 2018

Moving original netlist to extNetlist
Removing parasitic components from netlist
    presistors removed:  0
    pcapacitors removed: 0
    pinductors removed:  0
    pdiodes removed:     0
    trans lines removed: 0
    33 nodes merged into 33 nodes

Running netlist comparison program:  LVS
Begin comparison:    Oct 24 16:16:43 2018
@(#)$CDS: LVS.exe_64 version 5.1.0-64b 04/27/2009 03:12 (cicamd10) $

The net-lists match.
```

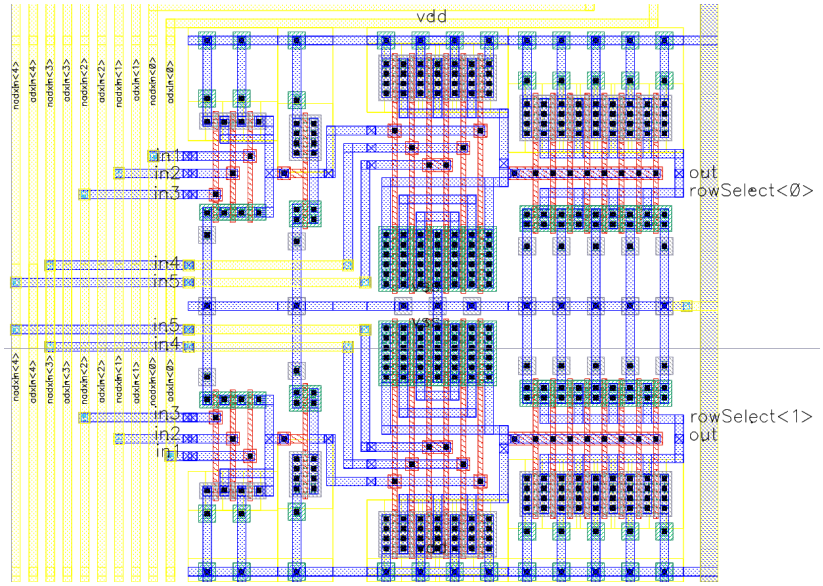
```
The net-lists match.

                    layout  schematic
                  instances
    un-matched        0   0
    rewired           0   0
    size errors       0   0
    pruned            0   0
    active           36  16
    total            36  16

                     nets
    un-matched        0   0
    merged            0   0
    pruned            0   0
    active           15  15
    total            15  15

                  terminals
    un-matched        0   0
    matched but
    different type        0   0
    total             8   8
End comparison:      Oct 24 16:16:44 2018


Comparison program completed successfully.
```

Full Decoder Layout, Full Decoder (Inv V adx line buffers)

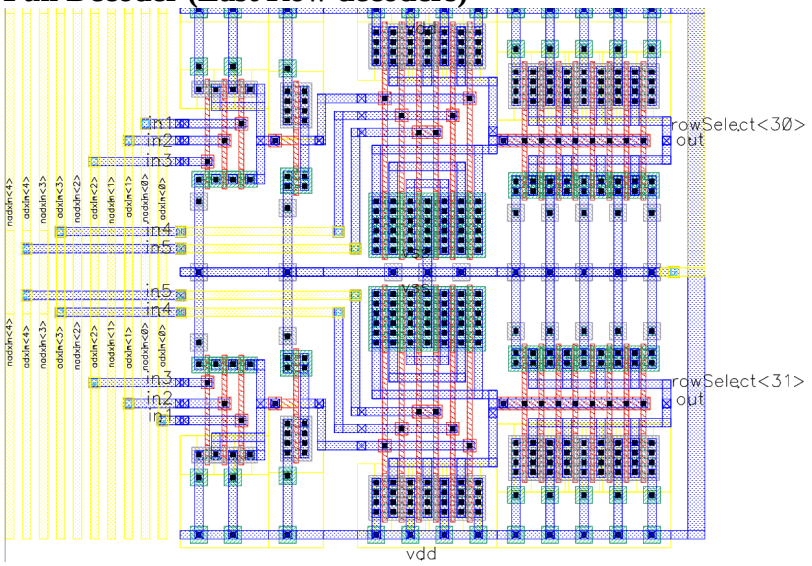




Full Decoder (Initial Row decoders)



Full Decoder (Last Row decoders)



Full Decoder Reports (DRC, ERC, LVS)

```
********   Summary of rule violations for cell "fullDecoderLayout layout"   ********
   Total errors found: 0
t
```

```
Job                                                                    /ERC' that was started at 'Oct 25 00:06:28 2018' has succeeded
```

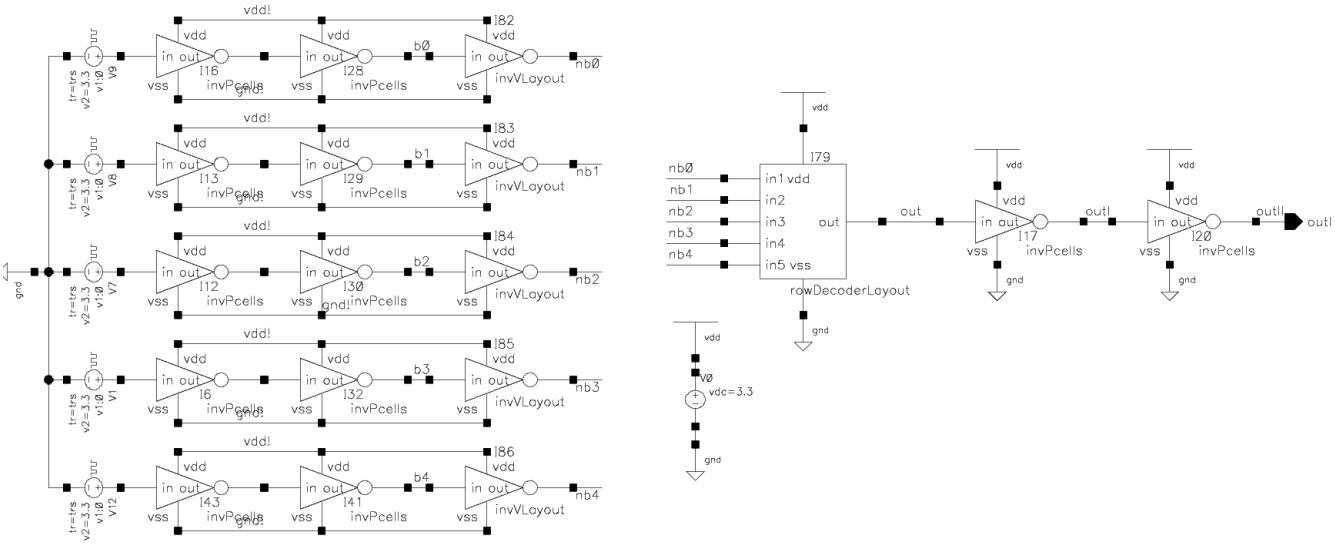
```
Begin netlist:    Oct 25 00:07:39 2018
    view name list  = ("auLvs" "extracted" "schematic")
    stop name list  = ("auLvs")
    library name    = "project_2"
    cell name   = "fullDecoderLayout"
    view name   = "extracted"
    globals lib = "basic"
Running Artist Flat Netlisting ...
End netlist:    Oct 25 00:07:40 2018

Moving original netlist to extNetlist
Removing parasitic components from netlist
    presistors removed:  0
    pcapacitors removed: 0
    pinductors removed:  0
    pdiodes removed:     0
    trans lines removed: 0
    278 nodes merged into 278 nodes


Begin netlist:    Oct 25 00:07:40 2018
    view name list  = ("auLvs" "schematic")
    stop name list  = ("auLvs")
    library name    = "project_2"
    cell name   = "fullDecoder"
    view name   = "schematic"
    globals lib = "basic"
Running Artist Flat Netlisting ...
End netlist:    Oct 25 00:07:41 2018

Moving original netlist to extNetlist
Removing parasitic components from netlist
    presistors removed:  0
    pcapacitors removed: 0
    pinductors removed:  0
    pdiodes removed:     0
    trans lines removed: 0
    1212 nodes merged into 1212 nodes

Running netlist comparison program:  LVS
Begin comparison:    Oct 25 00:07:41 2018
@(#)$CDS: LVS.exe_64 version 5.1.0-64b 04/27/2009 03:12 (cicamd10) $

The net-lists match.
```

```
The net-lists match.

                 layout  schematic
                instances
    un-matched      0   0
    rewired         0   0
    size errors     0   0
    pruned          0   0
    active          1172    532
    total           1172    532

                  nets
    un-matched      0   0
    merged          0   0
    pruned          0   0
    active          278 278
    total           278 278

                terminals
    un-matched      0   0
    matched but
    different type      0   0
    total           44  44
End comparison:      Oct 25 00:07:41 2018


Comparison program completed successfully.
```

3.2 Simulate the extracted view to obtain the values for the delay as in question 2.2 and for power dissipation as in questions 2.3, 2.4, 2.5.

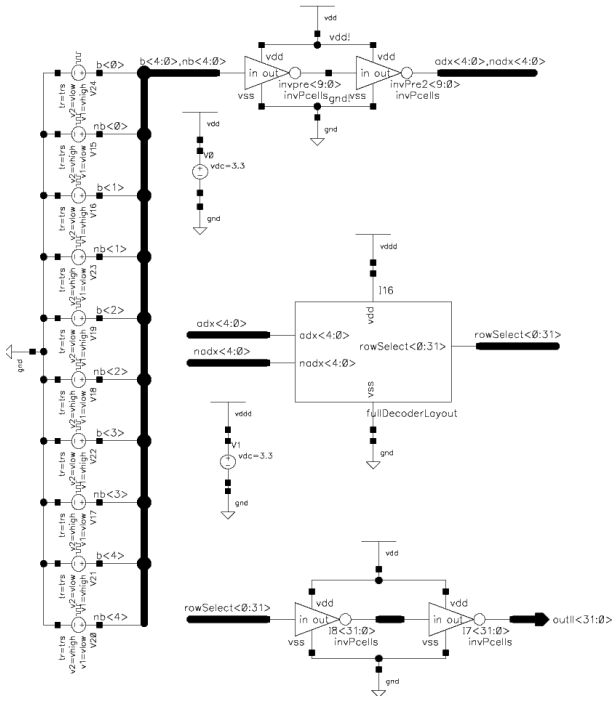
The post-layout simulation strategy outlined at the beginning of this section was followed, the row decoder testbench can be seen below:

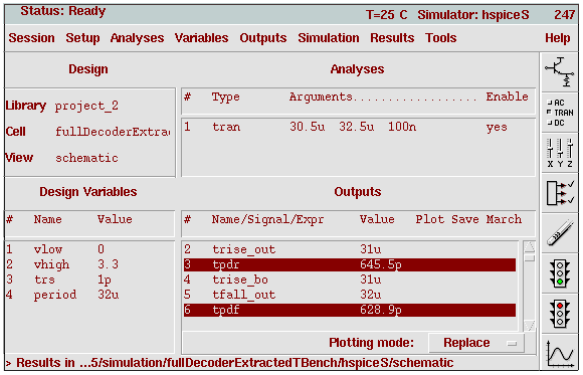
Post-layout simulation: Row-Decoder Testbench



We used this testbench to assess the functionality of our row-decoder extracted layout to make sure it was working before moving on to the full decoder.

Post-layout simulation: Full-Decoder Testbench Schematic with path delay results

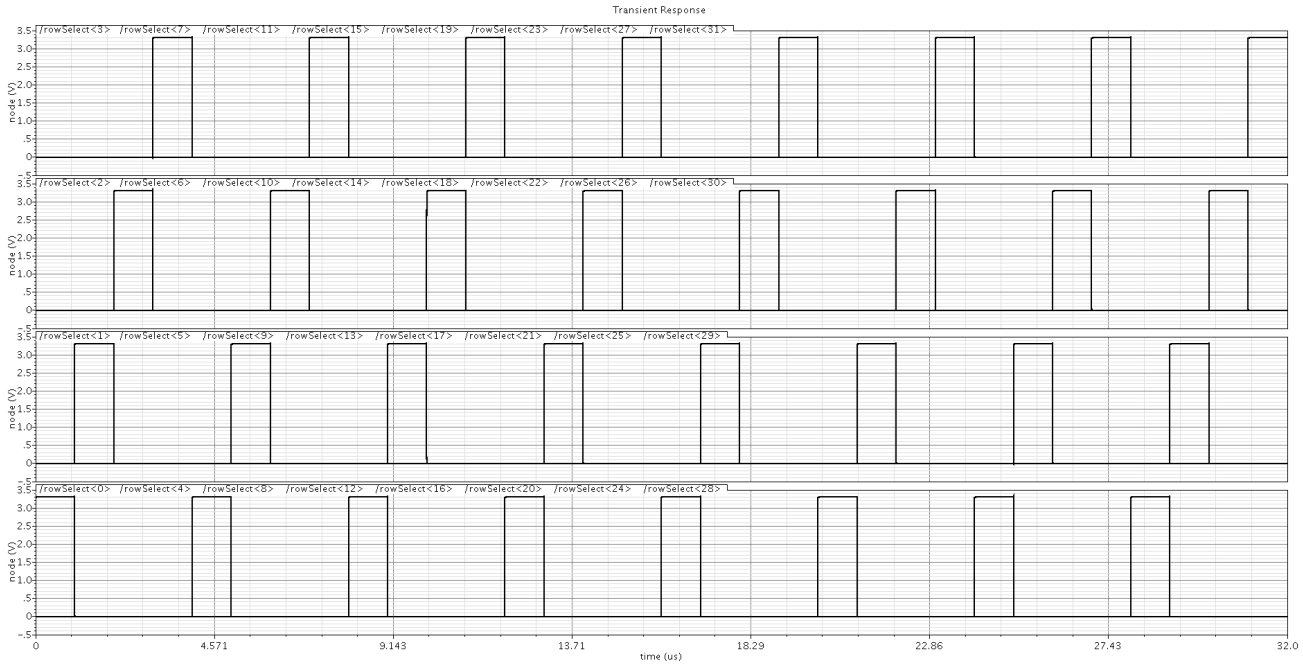




Rising propagation delay: 645.5 pS
Falling propagation delay: 628.9 pS

Average propagation path delay: 637.2pS

Additionally, the full decoder functionality was tested post-layout simulation, transient results over one address run from 0 to 31 can be seen below (overlaid on one plot as before):



3.3 Compare the results for delay and average dynamic power dissipation for your calculations

Therefore our average propagation path delay from our post-layout simulation is almost half (61.2%) of our simulated path delay, (results summarized at the end of this section). This could be due to many

reasons including: our re-sizing of gate transistors to ease multiplier and fingering strategies and potentially our layout extraction could have benefited from a “parasitics” switch to model the conductors capacitances in the layout (not just that of our active devices)

Path delay (estimated, simulated path delay, post-layout simulation path delay)

Row Decoder Path	Calculated D (τ)	τ (S)	Calculated D (S)	Measured tpdr	Measured tpdf	Measured Path Delay
	Path Delay	τ for tsmc 350nm	Path Delay in S	Path propagation rising delay	Path propagation falling delay	Average rise and fall Propagation delay
Estimate and Simulation	28.284	4E-11	1.13E-09	1.374E-09	7.081E-10	1.04E-09
Post Layout Simulation				6.455E-10	6.289E-10	6.37E-10

For the power consumption, the results are somewhat different from our expectations: the dynamic power consumption is half of our estimated value and more than half of our simulated value, furthermore the leakage static current consumption is significantly greater than we expected.

Power consumption	Total Current Consumption	Total Power Consumption	Static Current Consumption	Static Power Consumption	Dynamic Power Consumption
Estimate					1.56E-05
Estimate and Simulation	8.88E-06	2.93E-05	1.09E-08	3.61E-08	2.93E-05
Post layout simulation	2.47E-06	8.16E-06	4.35E-07	1.44E-06	6.73E-06

We foresee more accurate power measurements could be obtained via obtaining a “parasitics included” layout extraction – including metal and diffusions parasitic capacitances - which could add additional node capacitances to be driven by gates upon switching and could perhaps increase our power consumption to match our figures obtained in earlier sections.