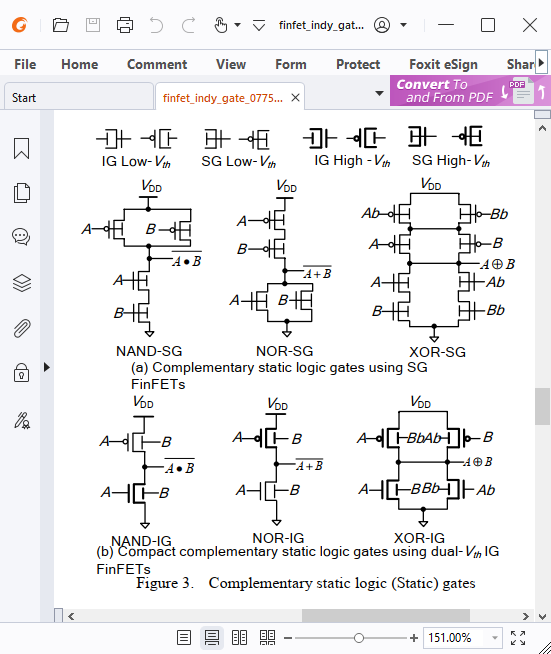
Problem-by-problem Scores (grader use)

|  |  |  |
| --- | --- | --- |
| Problem Number | Score | Totals |
| 1a | /12 | ============= |
| 1b | /13 | /25 |
| 2a | /12 | ============= |
| 2b | /13 | /25 |
| 3a | /12 | ============= |
| 3b | /13 | /25 |
| 4a | /12 | ============= |
| 4b | /13 | /25 |
| 5a | /12 | ============= |
| 5b | /13 | /25 |
| 6 | /25 | /25 |
| 7 | /13 | /25 |
| 8a | /12 | ============= |
| 8b | /13 | /25 |
| Total |  | /200 |

**1.** Figure depicts an XOR2 logic circuit built out of finFETs operating in **single gate** mode. (Schematic topology is same as for planar transistors.)

**1a**. To make it into a **2:1 mux** with data inputs A and B and control input SEL, such that:

if(SEL==1) Y(out) = D0; else if(SEL==0) Y(out) = D1, which inputs, D0, D1, SEL, SELB, connect to which ports? Put an X in each appropriate square.



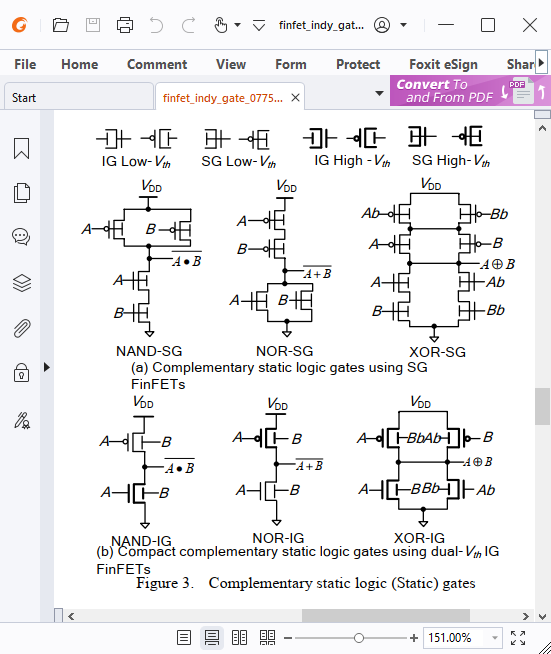
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR inputs | | SEL | SELB | D0 | D1 |
| PMOS | Ab |  |  | **X** |  |
| PMOS | A | **X** |  |  |  |
| PMOS | Bb |  | **X** |  |  |
| PMOS | B |  |  |  | **X** |
| NMOS | A |  | **X** |  |  |
| NMOS | B |  |  | **X** |  |
| NMOS | Ab | **X** |  |  |  |
| NMOS | Bb |  |  |  | **X** |

Example: If you to connect SEL to the upper left PMOS Ab port, check the upper left box in the grid above.

**NOTE: This actually builds an inverting MUX, with Y = !D0 or Y = !D1**

**Accept answer above or argument for inverting D0 and D1 at the input.**

**1b**. Now build an **XNOR2** logic gate using four finFETs operating in **independent gate** mode. Also label each device H for high threshold, or L for low threshold.



**All four are in high threshold mode. DIagram is XOR. For XNOR, students need to swap either As with Abs or Bs with Bbs.**

**2a**. Design a ganged 2-input NOR gate (two inverters in parallel, separate inputs, common output) with mobility ratio N/P = M and strength ratio\* = S. Size transistors such that worst case net gu = 1.

\*S = ratio of NMOS resistance to PMOS resistance or PMOS resistance to NMOS resistance, as appropriate.

PMOS width / NMOS width = **M/S**

gu / gd = **(S-1)/2**

pu / pd = **(S-1)/2**

**2b**. Repeat for a ganged 2-input NAND (same architecture, same M, same or inverted strength ratio S, different transistor sizes), **again with worst case gu = 1**.

PMOS width / NMOS width = **MS/1**

gu / gd = **2/(S-1)**

pu / pd = **2/(S-1)**

**3a**. Consider a carry lookahead adder that adds two 16-bit numbers.



Express the worst-case propagation delays in terms of the following parameters:

tpdg, tpdp = prop. delays from data in (A, B) to G and P out, respectively, for each 4-bit block

tpds = prop. delay from data in (A, B) to sum (S) out for each 4-bit block

tpa, tpo = prop. delays of AND and OR gates, respectively

tpcc = total (cumulative) prop. delay from Cin to Cout

tpcc = **max(tpdg, tpdp+tpa) + tpo + 3\*(tpa+tpo)**

tpd (Cin to S[16]) = **max(tpdg, tpdp+tpa) + tpo + 2\*(tap+tpo) + tpcs**

tpd (A[1] to S[16]) = **same as Cin to S[16]**

**Rubric:**

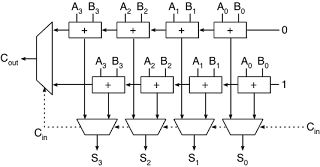
**3a)**

**TOTAL: 4 points for each question**

**i, ii) 1 per correct term**

**iii) Gave full 4 points if Tpd is same as prev (or follow grading rubric from ii)**

**3b**. Consider a 4-bit carry-select adder, as shown in this diagram.



Assuming Cin, A, and B arrive simultaneously, what is the worst-case propagation delay for Cin to Cout? Assume that **tpms** = select (here, carry) input to data output propagation delay and **tpmd** = data in to data out propagation delay of a 2:1 mux, and that **tpad** = data in to data out delay and **tpac** = data in to carry out delay of each full adder cell.

tpd (Cin to Cout) = **4\*tpac + tpmd**

**Total: 6 points**

**Rubric: 3 points for each correct term (gave partial points depending on coefficients)**

tpd (Cin to S[3]) = **3\*tpac + tpad + tpmd**

**TOTAL 7 points:**

**Rubric: 2 points for each correct term (gave partial points depending on coefficients)**

**Everyone got free point**

**4**. This quarter we have seen several memory address decoders which translate an N-bit-wide binary address pointer into 2N word lines. A binary **encoder**, used in Content Addressable Memories and other applications, performs the opposite function. Design a priority to **Gray** encoder which performs the following function:

|  |  |  |
| --- | --- | --- |
| **input** (W[3:0]) | **output** (A[1:0]) | **output** NULL (W=0) flag |
| 0000 | 00 | 1 |
| 1XXX | 00 | 0 |
| 01XX | 01 | 0 |
| 001X | 11 | 0 |
| 0001 | 10 | 0 |

**Assume W[3:0] and ~W[3:0] (complements) are available.**

**4a**. Boolean equations, as functions of W[x]:

A[1] = **!W[3]&!W[2]&(W[1]|W[0])**

A[0] = **!W[3]&(W[2]|W[1])**

NULL = No Match flag (in a CAM, for example) = **!W[3:0]**

**4b**. Design 4a equations in hardware, using only NAND2, NOR2, and NOT gates

If writing a netlist, use NAND(A,B) or !(A&B) to denote a NAND2 with inputs A and B, etc.

You may draw a schematic, instead. Either format for your solution is equally acceptable.

Assume again that inputs Wx and their complements are available (complimentary 😊 ) to you.

A[1] = **!W[3]&!W[2]&(W[1]|W[0])**

= **!(W[3]|W[2]) & !(!W[1]&!W[0])**

= **!((W[3]|W[2]) | (!W[1]&!W[0]))**

A[0] = **!(W[3] | (!(W[2]|W[1]))**

NULL = **!(!(!W[3]&!W[2]) | !(!W[1]&!W[0]) )**

**5**. Start with an N-bit adder module which performs the operation {cout, sum}=a + b + cin, where sum, a, and b are N-bit **unsigned** numbers. 

**5a**. Add saturating logic (2:1 muxes, NAND2, NOR2, or XOR gates, etc.) that replaces S with the largest possible representable value in the event of an overflow. You may draw a schematic or write a Boolean expression.

**if(Cout) S[N:1] = N’b1;**

**else S[N:1] = S[N:1]**

**Rubric:**

**TOTAL: 12 Points  
GAVE FULL POINTS FOR CORRECT EXPRESSION/SCHEMATIC (Most people got this correct)**

**If close to correct, I gave 10 points**

**5b**. Repeat 5a for **two’s complement** A, B, and S, replacing S with a maximum positive value for positive overflow, or most negative value for negative overflow.

**if(A[N]&B[N]&!S[N]) S = {1’b0,{(N-1){1’b1}}};**

**else if(!A[N]&&!B[N]&S[N] S = {1’b1,{{(N-1){1’b0}}};**

**else S[N:1] = S[N:1];**

**Mux: positive overflow 🡪 0111…1; neg overflow 🡪 1000**

**TOTAL: 13 Points  
GAVE FULL POINTS FOR CORRECT EXPRESSION/SCHEMATIC (Most people got this correct)**

**If close to correct, I gave 11 points**

**6**. Develop the delay model of a **wordline decoder** for a RAM with 2r rows and 2c columns. Assume the input capacitance = that of a column, so **H = 2c** if we have unit-sized access transistors. Assuming true and complementary binary address inputs are available and using static CMOS logic, including invereters and r-input NAND or NOR gates, express wordline decoder logical effort and branch factor as functions of r and c. Assume n/p = M and predecoding is not used.

G = **(r+M)/(1+M)** B = **2r-1** F = **(r+M)/(2(1+M)) \* 2r+c**

RUBRIC:

**Total 25 points.**

**8 points in total for G (2 points per correct term)**

**8 points for correct B (+4 for decent attempt, +2 if attempt was made)**

**9 points for correct F (Or if GBH was calculated correctly based on their G and B values)**

**7**. Show the partial and final products for Radix-4 Booth multiplication of **unsigned** operands Y\*X = 100111 \* 011001.

Hint: Use this table and start with X[-1] = 0, then X[5:0] = operand, end with X[7:6] = 00.



Partial Product 0 = \_**1**\_\_\*Y (Answer = -2, -1, 0, 1, or 2)

Partial Product 1 = \_**-2**\_\*Y (Answer = -2, -1, 0, 1, or 2)

Partial Product 2 = \_**2**\_\_\*Y (Answer = -2, -1, 0, 1, or 2)

Partial Product 3 = \_**0**\_\_\*Y (Answer = -2, -1, 0, 1, or 2)

Final answer = PP3\*64 + PP2\*16 + PP1\*4 + PP0\*1 = \_**975**\_\_ (decimal)

X = \_**25**\_\_\_ (decimal) Y = \_\_**39**\_\_\_ (decimal)

**8a**. Assuming a 4:7 SEC-SED Hamming FEC, complete each of the following. Assume no errors for 8a.

Format: D3 D2 D1 C2 D0 C1 C0, where

Constraints: D3^D1^D0^C0=0 D3^D2^D0^C1=0 D3^D2^D1^C2=0

**TOTAL 12 points**

For D[3:0] = {0, 1, 1, 1}, **C2 = 0 C1 = 0 C0 = 0**

**RUBRIC: 1 point for each correct value**

For D[3:0] = {1, 0, 1, 1}, **C2 = 0 C1 = 0 C0 = 1**

**RUBRIC: 1 point for each correct value**

For C2 = 0, C1 = 1, C0 = 0, D3 = \_\_**1**\_\_ D2 = \_\_**0**\_\_ D1 = \_**1**\_\_\_ D0 = \_\_**0**\_\_

or: D3 = \_\_**1**\_\_ D2 = \_\_**1**\_\_ D1 = \_**0**\_\_\_ D0 = \_\_**1**\_\_

**RUBRIC: 0.75 points for each correct value**

**8b**. For the following 7-bit sequences of D3 D2 D1 C2 D0 C1 C0, each of which may have 0 or 1 inverted (erroneous) bits, write the **number of** check bit errors (0 or 1), the number of data errors (0 or 1) and the recovered D values, corrected if necessary.

**TOTAL 12 points**

0111101 Check Bit Errors: \_**0**\_\_\_ Data Errors: \_**1**\_\_\_ Correct Message D[3:0] = \_\_**0101**\_\_\_\_\_\_\_

1001011 Check Bit Errors: \_\_**0**\_\_\_ Data Errors: \_**0**\_\_\_ Correct Message D[3:0] = \_\_**1000**\_\_\_c

0101000 Check Bit Errors: \_**1**\_\_\_ Data Errors: \_\_**0**\_\_ Correct Message D[3:0] = \_\_\_**0100**\_\_\_\_

**RUBRIC: 1 point for each check bit error and data errors. 2 points for correct message (no partial credit)**