University of California, San Diego

M.S. Exam: Computer Engineering

Spring 2014

Name: _____

Instructions:

All work to be done on the attached sheets. Write your name at the top of every sheet. 25 points/problem.

1. An AOI22 standard logic gate executes: Y = !((A & B) | (C & D)), i.e., the logical equivalent of two AND gates feeding a NOR gate. (In Verilog, & = AND, | = OR, ! = NOT.)

a. Design one in static CMOS logic which minimizes the number of transistors.

W&H 4th ed. Fig 1.18e

b. Show the sizes of the transistors needed to obtain the same drive as a unit inverter. (Assume P mobility is 1/2 that of N.) **P = 4, N = 2**

c. Compute the logical effort of each of the four inputs. **(4 + 2 = 6)/3 = 2**

d. Provide a rough estimate of propagation delay, based on total output diffusion capacitance.

d = C_{out}/x + p, where x = C_{in}/g = 3 and p = 12/3 = 4, so d = 4+4 = 8 in RC units

2. Use the following timing parameters and consider a combinational logic circuit which sits between two flip-flops or two-latches.

	t_{setup}	clk-to-Q delay	D-to-Q Delay	contamination delay	t_{hold}	Clock Cycle	clock pulse width
Flip-Flop	65ps	50ps		35ps	30ps	500ps	80ps
Latch	25ps	50ps	40ps	35ps	30ps	500ps	80ps

a. What is the maximum allowable logic propagation delay between two flip-flops, with zero clock skew?

$t_{cy} - t_s - t_{pcq}$ = 500 - 65 - 50 = 385 ps W&H 4th ed, equn 10.1

b. Repeat a for two pulsed latches instead of two flip-flops.

$t_{cy} - t_{pdq}$ = 500 - 40 = 460 ps because $t_{pdq} > t_{pcq} + t_s - t_{pw}$ W&H 10.6

c and **d**. Repeat a and b, respectively, with a clock skew between the two flip-flops or latches of +/- 50ps.

c. **min. t_{cy} = 500 - 100 = 400 ps, so t_{pd} = 385 - 100 = 285 ps** W&H 10.12

d**. t_{pd} = 500 - t_{pcq} – t_s + t_{pw} – t_{sk} = 500 - 50 - 25 + 80 - 100 = 405 ps** W&H 10.17

M. S. Exam: Computer Engineering

3. Starting with a group of conventional 1-bit full adders, design a comparator accepts two N-bit binary numbers and returns: a > b, a = b, and a < b when:

a. a and b are unsigned

b. a and b are two's complement

c. a is unsigned, but b is two's complement

You may sketch a schematic of your solution and/or write it in structural gate-level [System]Verilog or VHDL. You do not need to show how the 1-bit full adders themselves are made.

a. {c, s[N-1:0]} = a[N-1:0] + ~b[N-1:0] + 1

aeqb = !s

altb = ~c

agtb = c && s

b. {c, s[N-1:0]} = {~a[N-1],a[N-2:0]} + {b[N-1],~b[N-2:0]} + 1

The three flag equations themselves are unchanged.

c. {c, s[N-1:0]} = ~a[N-1:0] + b[N-1:0] + 1

aeqb = !s

agtb = c || b[3];

altb = !agtb && !aeqb

4. A 4:7 Hamming forward error correction (FEC) encoder generates three parity bits, C[2:0],from four data bits, D[3:0], using the formula: C[0] = D[3]^D[1]^D[0], C[1] = D[3]^D[2]^D[0], C[2] = D[3]^D[2]^D[1].

a. What is the Hamming distance between any two 7-bit codes so generated?

Invert D[3] → HD = 4, because all C's invert, as well

Invert D[2], D[1], or D[0] → HD = 3, because two C's invert, as well

So **min. HD = 3**

b. Design the corresponding decoder. You may write Verilog or VHDL code or draw a schematic.

Let c, d = parity and data, as received; cr = parity reconstructed from d; dr = corrected d

Let cx = cr ^ c = syndrome

cr[0] = d[3]^d[1]^d[0], cr[1] = d[3]^d[2]^d[0], cr[2] = d[3]^d[2]^d[1].

dr[3] = d[3] ^ (cx==3'b111)

dr[2] = d[2] ^ (cx==3'b100)

dr[1] = d[1] ^ (cx==3'b010)

dr[0] = d[0] ^ (cx==3'b001)

c. How many errors can you design detect, and how many can it correct?

Can detect 2 errors, correct only 1

d. Verify your design by showing what happens if D[3] arrives inverted, whereas D[2:0] and C[2:0] all arrive correctly.

If d[3] = ~D[3], then c[2:0] = ~C[2:0], so cx = 3'b111, which points to D[3]