# EEM16 Midterm

Nate Sookwongse

TOTAL POINTS

**59 / 65**

QUESTION 1

## 1 Problem #1 18 / 18

- ✓ **- 0 a) is Correct**
- ✓ **- 0 b) is correct**
- ✓ **- 0 c) is correct**
- ✓ **- 0 d) is correct**
- ✓ **- 0 e) is correct**
- - 1 a) added don't cares as minterms
- - 3 a) is incorrect
- - 2 b) is partially correct
- - 5 b) is incorrect
- - 3 c) is incorrect
- - 3 d) is incorrect
- - 2 e) is partially incorrect
- - 4 e) is incorrect
- - 1 d) partially incorrect
- - 2 c) partially incorrect

QUESTION 2

## 2 Problem #2 10 / 14

- - 0 all correct
- ✓ **- 2 Part (a): answer other than 2, 3 or 4**
- ✓ **- 1 Part (b): deduct for wrong demorgan**
- - 0.5 Part(b): partial improper SoP
- ✓ **- 1 Part(b): For improper SoP form**
- - 2 Part(b) quite wrong SoP
- - 4 Part(b): all wrong
- - 1.5 Part(c): if Not distributive
- - 2 Pat(c) no 6 SoP
- - 1 Part(c). For each wrong SoP term
- - 4 Part(c): all wrong
- - 0.5 Part(d): Partial wrong reduced expression
- - 1 Part(d): quite wrong reduced expression
- - 1 Part(d): For one missing property
- - 2 Part(d): For two missing properties

QUESTION 3

## 3 Problem #3 22 / 22

- ✓ **- 0 Correct. Good Job.**
- - 1 (a) math error
- - 2 (a) incorrect
- - 1 (a) (b) or (c) math error
- - 2 (b) incorrect
- - 2 (c) incorrect = -105
- - 0.5 (c) negative error for 2's complement
- - 2 (d) incorrect hex
- - 2 (e) incorrect BCD
- - 1 (f) partial credit
- - 2 (f) incorrect
- - 2 (g) incorrect bias
- - 2 (g) incorrect real number
- - 1 (g) partial credit for error
- - 6 (h) incorrect
- - 2 (h) incorrect floating point choice
- - 1 (h) partially incorrect min bits for mantissa
- - 2 (h) incorrect min bits for mantissa
- - 1 (h) partially incorrect min bits for exponent
- - 2 (h) incorrect min bits for exponent

QUESTION 4

## 4 Problem #5 9 / 11

- - 0 (a) Correct. Nice Job!
- - 7 (a) incorrect
- - 5.5 (a) partial for ok attempt
- - 3 (a) Too much excessive logic
- - 0 (a) No or incorrect Hit logic
- ✓ **- 0.5 (a) Some unnecessary logic**
- - 0 (b) Correct. Nice Job!
- - 4 (b) incorrect
- - 3.5 (b) Partial credit for attempt
- - 1 (b) incorrect bit weight (hop and adder cnt)
- - 1 (b) no or incorrect adder count or hop

- **0.5** (b) Good attempt but slightly too many FA

✓ - **1.5** (b) ok attempt but incorrect design

- **3** (b) partial credit for attempt

## Question #1

Consider the Boolean function defined by the truth table below where *A, B, C, and D* are inputs, and *Y* is the sole output.

| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | X |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | X |

✓(a) Complete the following statements

$$Y = \sum m(\ 0, 4, 5, 6, 7, 8, 9, 11, 14\ )$$

✓(b) Complete the Karnaugh Map shown below, **circle** the prime implicants.

B

**AB** A

D C **CD**

| | "00" | "01" | "11" | "10" |
|---|---|---|---|---|
| "00" | 1 | 1 | 0 | 1 |
| "01" | 0 | 1 | X | 1 |
| "11" | 0 | 1 | X | 1 |
| "10" | 0 | 1 | 1 | 0 |

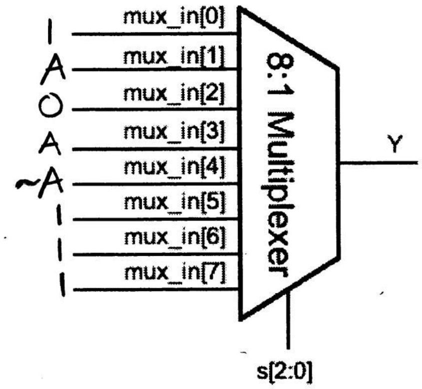
How many prime implicants are there? 7

✓(c) Write the Boolean (sum-of-product) expression for the essential prime implicants (if any).

EssentialPrimeImplicants = $(B \wedge C) \vee (A \wedge D)$

(d) Implement the function *Y* using an 8-input multiplexer. The select signal is *s[2:0]={B,C,D}* where *s=3'b100* is *B*=1 and *C*=*D*=0 selecting the input *mux_in*[4]. *A* or ~*A* are permissible as inputs, *mux_in*[7:0]. Write the desired inputs on the figure below.

| A | B C D | Y |
| --- | --- | --- |
| 0 | 0 0 0 | 1 |
| | 1 0 0 | |
| | 1 0 1 | |
| | 1 1 0 | |
| | 1 1 1 | |
| 1 | 0 0 0 | |
| | 0 0 1 | |
| | 0 1 1 | |
| | 1 0 1 | x |
| | 1 1 0 | |
| | 1 1 1 | x |



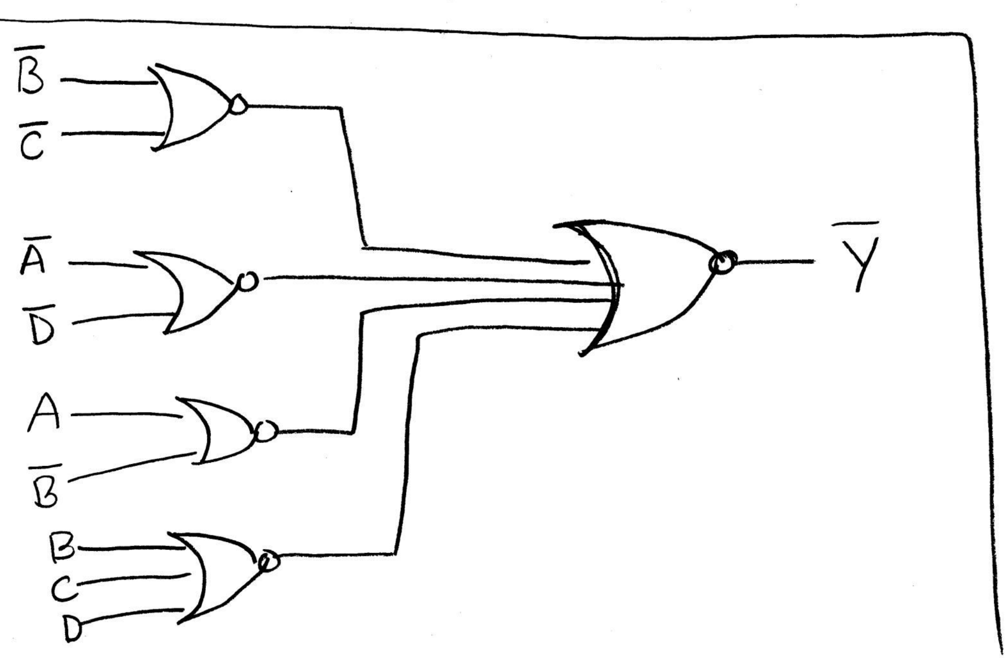

(e) Implement $\neg Y$ using the minimum # of NOR gates with fewest # of inputs (minimize literals and terms). Product of sums

$$Y = (B \wedge C) \vee (A \wedge D) \vee (\bar{A} \wedge B) \vee (\bar{B} \wedge \bar{C} \wedge \bar{D})$$

$$\bar{Y} = \overline{(B \wedge C)} \wedge \overline{(A \wedge D)} \wedge \overline{(\bar{A} \wedge B)} \wedge \overline{(\bar{B} \wedge \bar{C} \wedge \bar{D})}$$

$$\bar{Y} = (\bar{B} \vee \bar{C}) \wedge (\bar{A} \vee \bar{D}) \wedge (A \vee \bar{B}) \wedge (B \vee C \vee D)$$

$$\bar{Y} = \overline{\left[ \overline{(\bar{B} \vee \bar{C})} \vee \overline{(\bar{A} \vee \bar{D})} \vee \overline{(A \vee \bar{B})} \vee \overline{(B \vee C \vee D)} \right]}$$



## Question #2

$c \wedge \bar{d} \wedge \bar{e}$

$$Y = \neg\left(\neg(a \wedge \neg b) \vee \left(c \wedge \neg(d \vee e)\right)\right)$$

(a) For the above Boolean function, if you were to convert the above expression into a sum-of-product representation, how many times did you have to apply DeMorgan's theorem?

1

$(a \wedge \bar{b}) \wedge \neg(c \wedge \bar{d} \wedge \bar{e})$

$(a \wedge \bar{b}) \wedge (\bar{c} \vee d \vee e)$

(b) For part (a), what is the resulting function?

$Y =$ $\neg\left(\neg(a \wedge \neg b) \vee (c \wedge \neg d \wedge \neg e)\right)$

(c) The following expression can be written as a 6-term sum-of-product,

$$Y = (a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

$x * (a + b + c)$

$xa + xb + xc$

What Boolean property do you need to apply to do this?

Distributive property

Without reducing, what are the 6 product terms?

$((a \vee b) \wedge a) \vee ((a \vee b) \wedge \bar{b}) \vee ((a \vee b) \wedge \bar{c})$

$Y = (a \wedge a) \vee (a \wedge b) \vee (a \wedge \bar{b}) \vee (b \wedge \bar{b}) \vee (a \wedge \bar{c}) \vee (b \wedge \bar{c})$

(d) The 6-term sum-of-product of part (c) can obviously be reduced.
What is the reduced expression?

$Y = a \vee (b \wedge \bar{c})$

What Boolean axioms or properties are needed for the reduction?

| | |
|---|---|
| axiom → $a \wedge a = a$ | $b \wedge \bar{b} = 0$ ← axioms |
| properties → Combining $(a \wedge b) \vee (a \wedge \bar{b}) = a$ | $a \vee a = a$ ← |
| properties → absorption $a \vee (a \wedge \bar{c}) = a$ | |

$Y = a \vee a \vee 0 \; (a \wedge \bar{c}) \vee (b \wedge \bar{c})$

$a \vee (b \wedge \bar{c})$

## Question #3

(a) The following 8 bits can be used to represent different numbers depending on the encoding

8b'10010111 1001 0111

If this was unsigned, what is the corresponding integer? 151

$2^7$ $2^4$ $2^2 2^1 2^0$

(b) If the 8 bits in (a) was sign magnitude, what is the corresponding integer? -23

− 001 0111

(c) If the 8 bits in (a) was 2's complement, what is the corresponding integer? -105

− 0110 1001 $2^0+2^3+2^5+2^6$

(d) If the 8 bits in (a) was hexadecimal, what is the corresponding hexadecimal? 97

1001 = 9 0111 = 7

(e) If the 8 bits in (a) was binary coded decimal, what is the corresponding integer? 97

(f) If the 8 bits is fixed point 1001.0111, what is the corresponding number? 9.4375

$2^3+2^0+2^{-2}+2^{-3}+2^{-4}$

(g) If the 8 bits in (a) was a 4E3 floating point number (IEEE format S+EEE+MMMM),

What is the bias? 3 $2^{k-1}-1 = 2^2-1 = 3$

1 001 0111

What is the corresponding real number? -0.359375

$M = 1 + 2^{-2}+2^{-3}+2^{-4} = 1.4375$

$E = 1-3 = -2$

$\# = -M\,2^E$

(h) Military temperature range is -55°C to +125°C with 1% accuracy.

Would you choose floating point or fixed point? floating point

If you are to represent this in floating point, what is the minimum # of bits for mantissa? 6

$0.25°C \times 0.01 = \underline{0.0025}$

frac 8 bits

$2^{-8}$

And, what is the minimum # of bits for exponent? 4

$2 \times 2^E$

$E = 6$

exp = 4

bias = 7

.01 E = -2

$1.\underbrace{0000}_{0101}\underbrace{000000}$

$E = 6 - 7$

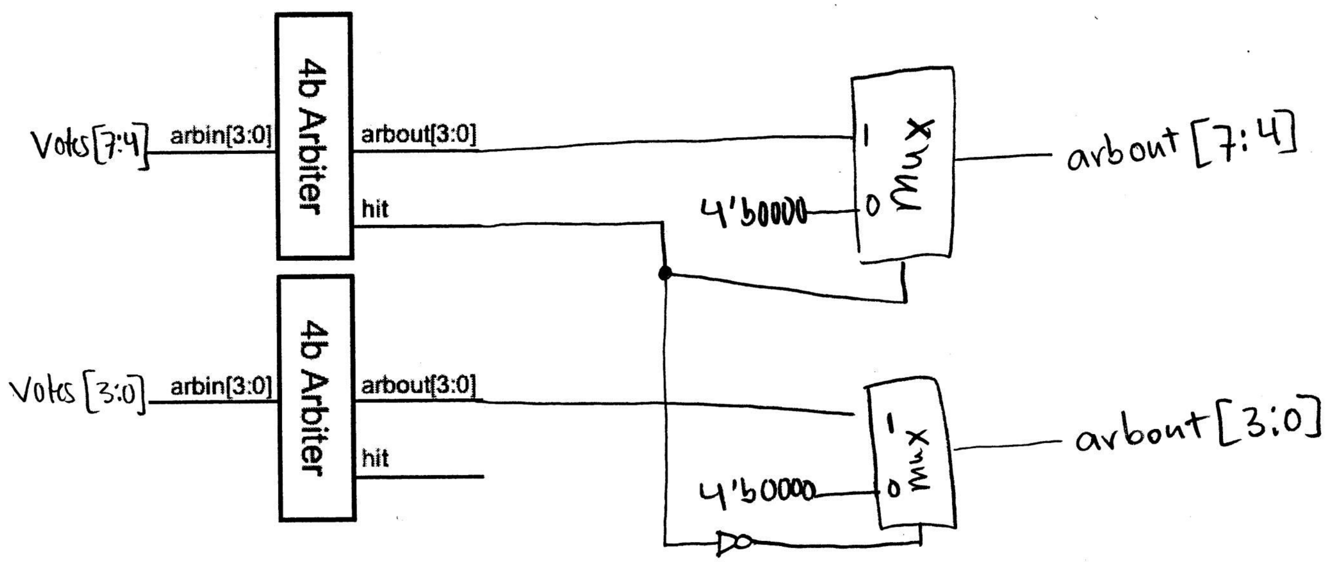
E = −2 − Bias

−2 − 7 =

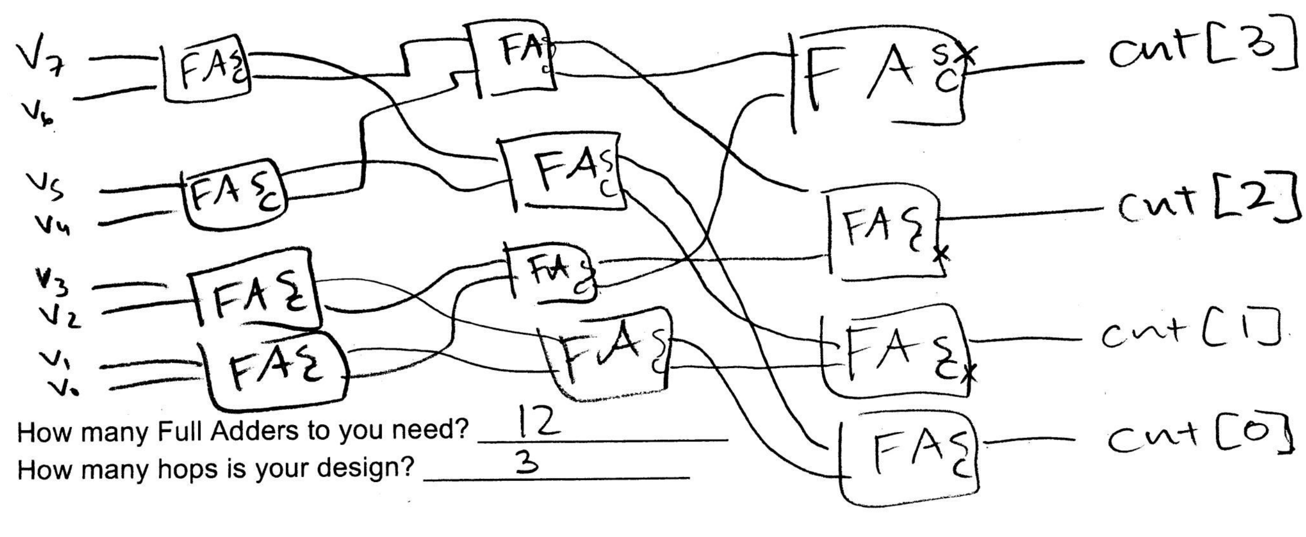
exp = −2 + Bias = 5

## Question #5

(a) Given 8-bit input, *votes*[7:0], in which any number of the inputs can be a 1'b1. Build an **arbiter** that provides an 8-bit output, *arbout*[7:0], that is 1-hot. The hot signal corresponds to the position with the highest priority. Note that votes[7] has higher priority than votes[6] etc. You have available to you a module ARB that is a 4-bit arbiter already built that you **must** use. ARB accepts as inputs *arbin*[3:0] and outputs *arbout*[3:0] and a *hit* signal to indicate that one or more of the signals is a 1'b1. You also have available to you INV (inverters), and 2-input MUX (multiplexers). Recall that you can implement considerable arbitrary logic with 2-input MUXs.



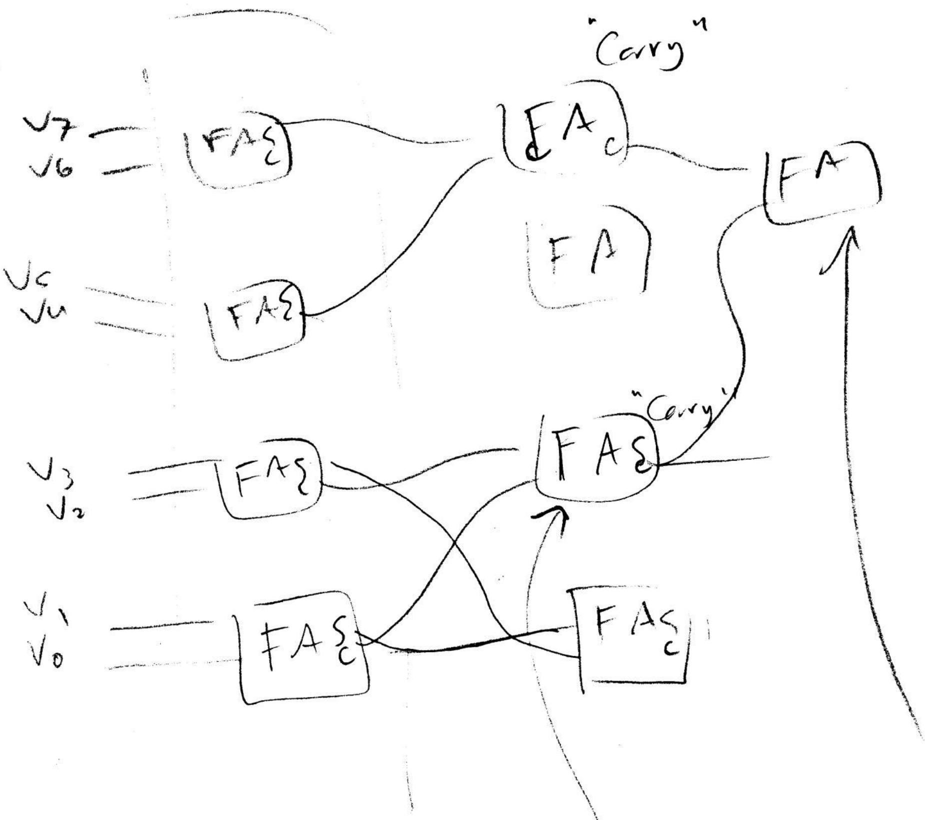

(b) Now, the *votes*[7:0] need to be counted. You have available Full Adders (FA) as building blocks for implementing a design. If the delay of the logic is determined by the number of hops where each hop is the traversal of a Full-Adder from any input (*a,b,and c*) to any output (*sum, carry*). Design your block to minimize this delay. Note that your design should output 4 bits to indicate the binary count, *cnt*[3:0].




How many Full Adders to you need? 12

How many hops is your design? 3

Blank page for scratch work




4 3 2 1

0000
0001 -
0010
0011 -
0100
0101 -
0110
0111 -
1000

C = 1 ⇒ 2 votes

C = 1 ⇒ 4 votes

C = 1 ⇒ 8 votes

Blank page for scratch work