Exam 1

Due No due date **Points** 125 **Questions** 5
Available Mar 8 at 7pm - Mar 8 at 9pm 2 hours **Time Limit** None

Instructions

Remember ::

A. **Once you hit "submit" you CANNOT return to the exam.** So you should hit "submit" only when you are sure you are done.
   - **Your exam will auto-submit at 9:00pm**

B. **For each question, you must mention HOW/WHY you got/chose the answer, and provide the reasoning you used to get your answer. If you do not do this, you WILL lose points.**
   - **If you give me more than one WHY/HOW for the same question, I will grade all your answers, and award you points for the answer which obtained the least points. So it's best to give me ONE answer.**

C. Questions marked [749 only]
   - ECEN 449 students: DO NOT answer these questions. You will receive these points for free.
   - ECEN 749 students: You MUST answer these questions. Your grade on these questions will count toward your final grade

D. TOTAL POINTS are 125 points for both ECEN 449 and ECEN 749 students

————————————————————Some constants/notation.
————————————————————

A. 1 millisecond = 10^{-3} sec
B. 1 microsecond = 10^{-6} sec
C. 1 nanosecond = 10^{-9} sec
D. 1 picosecond = 10^{-12} sec
E. 1 byte = 1 B = 8 bits
F. 1 kilobyte = 1 KB = 10^3 byte
G. 1 megabyte = 1 MB = 10^6 byte
H. 1 gigabyte = 1 GB = 10^9 byte
I. 1 cm = 10^{-2} m
J. The speed of light is c = 3 x 10^8 m/s
K. Use "**abar**" to refer to the complement of logical variable "**a**"

This quiz was locked Mar 8 at 9pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	79 minutes	104 out of 125

Correct answers are hidden.

Score for this quiz: **104** out of 125
Submitted Mar 8 at 8:18pm
This attempt took 79 minutes.

Question 1

17 / 25 pts

ANSWER IN THE BOX BELOW. DO NOT SCAN/UPLOAD YOUR ANSWER

For this question, I want you to fill in the "K-map" for a gate, using the 4-valued algebra of Verilog. Let's call our gate INV-or-BUF. This gate has an input "in", an output "out", and a control signal (which is also an input signal) "control". When control=1, the gate behaves as an INV gate, and when control=0, the gate behaves as a BUF. I have included a picture of a blank "K-map" below.

In your answer, you need to provide the entries of the "K-map" below. Your answer should consist of 4 rows (one per line), with 4 values written in each row, representing the values in the first, second, third and fourth rows of the "K-map" respectively.

control

in	0	1	X	Z
0				
1				
X				
Z				

b) Suppose I have a gate which has two inputs and one output, each of which uses the 4-valued algebra of Verilog. What is the theoretical maximum number of such gates possible? Each unique assignment of values to the “K-map” (for the 4-valued algebra) is a separate gate.

c) Suppose I have a gate which has two inputs and one output, each of which uses a new 5-valued algebra that I define (with values (1, 0, X, Z0, Z1), where 1, 0, and X have their usual meaning, while Z0 and Z1 mean high-impedance-0 and high-impedance-1 respectively). What is the theoretical maximum number of such gates possible? Each unique assignment of values to the “K-map” (for the 5-valued algebra) is a separate gate.

Your Answer:

(a) {

0 1 X X

1 0 X X

X X X X

X X X X

}

(b) {

input 1 has 4 possibilities

input 2 has 4 possibilities

output has 3 possibilities (0, 1, X)

input 1 * input 2 * output = 4 * 4 * 3 = <u>48 possible gates</u>

}

(c) {

input 1 has 5 possibilities

input 2 has 5 possibilities

output has 3 possibilities (0, 1, X)

input 1 * input 2 * output = 5 * 5 * 3 = <u>75 possible gates</u>

}

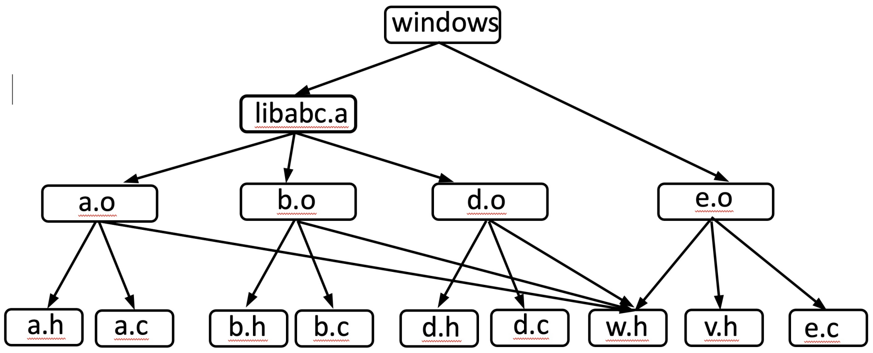
9 4 4 / 9 8 8

Question 2

20 / 25 pts

ANSWER IN THE BOX BELOW. DO NOT SCAN/UPLOAD YOUR ANSWER

Consider the graph that represents the file dependencies for my C-based software project. I ask my trusty assistant Pat to convert this graph into a Makefile.



Suppose I ran "make" at 6pm. The compilation finishes by 6:02pm.

a) What is the minimum number of targets that are rebuilt when exactly one input file (a single .c or .h file) is changed at 7pm?

b) What is the maximum number of targets that are rebuilt when exactly one input file (a single .c or .h file) is changed at 8pm?

c) When I change the file v.h at 9pm, and then do a "make", no files are recompiled. Should I fire Pat?

Your Answer:

(a) {

If "v.h" is changed at 7pm, then only "e.o" and "windows" are rebuilt.

Because the "v.h" file has the least number of dependents, the <u>minimum</u> number of targets rebuilt is <u>2 targets</u>.

}

(b) {

If "w.h" is changed at 8pm, then "a.o", "b.o", "d.o", "e.o", "libabc.a", and "windows" are rebuilt.

Because "w.h" has the most dependents, the maximum number of targets rebuilt is 6 targets.

}

(c) {

Firing Pat seems a bit drastic. Since "v.h" does not have any C-file dependents, no C-files will be recompiled. However, if necessary, the make utility should regenerate "e.o" and "windows".

}

9 9 2. / 9 9 7

Question 3

25 / 25 pts

ANSWER IN THE BOX BELOW. DO NOT SCAN/UPLOAD YOUR ANSWER

Each part of this question is independent of the other.

a) The company I am working for is trying to decide whether we should enter a new business. The new business will develop 3D headsets for Virtual Reality (VR). My boss puts me in charge of this business division. She wants us to have a first version of the product to market in 6 months, and she does not mind if the first product is heavy/slow/bulky. She says it's fine if we make a loss with this first version of the product. She expects we will have a much larger demand (100s of millions of units sold) after 2 years, so we should have a second version of the product by then, which implements a streamlined/fast/light headset. We made a prototype of this product in software, and it rendered the 3D VR images with a great deal of glitches and lag. What platform (software or FPGA or ASIC or custom IC) should I use for the first and second versions of the product?

b) Suppose I have a Verilog design which has 249 "reg" declarations. If I compile this Verilog file, how many flip-flops will be inferred?

c) We discussed in class that the meaning of "always @(x or y)" is -- "*whenever there is an event on x, or an event on y*". Suppose, instead, that the meaning was "*whenever x is high OR y is high*". What would be the semantic problem with this meaning of "always @(x or y)"?

Your Answer:

(a) {

For the <u>first version</u> of the product, we should use an <u>FPGA</u> platform. It will reduce the glitches and lag generated by the software approach, while still being quick to develop.

For the <u>second version</u> of the product, <u>ASIC</u> would be a better approach. Nowadays, a purely custom IC product is unreasonable, especially for a highly complex system like VR. ASIC will take a little bit more time than FPGA, but since we have a longer deadline and higher standards to meet, this seems like the better option.

}

(b) {

"reg" declarations do not directly result in inferred flip-flops. Inferred memory is dependent on other things in the code. (i.e. incomplete assignments)

}

(c) {

<u>The always block would not trigger when either "x" or "y" went from high to low</u>. Also, as long as "x" or "y" continued to be high, the always block would run continuously. The entire purpose of doing "always @(x or y)" is

to trigger on any change, assumably to update some values that depend on x or y.

}

9 7 9 / 9 7 9 Ncie

Question 4

17 / 25 pts

ANSWER IN THE BOX BELOW. DO NOT SCAN/UPLOAD YOUR ANSWER

Consider a computer program, which consists of 1,000,000 instructions. Each instruction is one byte long. These instructions run on a CPU which operates with a clock frequency f = 1GHz. The computer has a traditional memory hierarchy as we discussed in class (with a cache, a DRAM and an SSD). The latencies of the cache, DRAM and SSD are 1, 200, and 10^6 clock cycles respectively. (Note -- the SSD latency being 10^6 means that the first byte takes 10^6 cycles to arrive from SSD to DRAM, after it is requested. However, subsequent bytes take only one additional cycle to arrive).

When I run the program, the first thing that happens is that the entire program is loaded from SSD to DRAM. Then, after the program has been fully loaded, each of the instruction of the program is run. When each instruction is run, there is a probability P that it is found in the cache, in which case it would take a latency of 1 clock cycle to fetch the instruction from the cache. If the instruction is not found in cache, it must be fetched from DRAM. Running any instruction after it is fetched takes one cycle.

a) What is the total time it takes to run the program if P = 90% ?

b) What is the total time it takes to run the program if P = 95%?

c) What is the theoretical minimum total time it takes to run the program?

Your Answer:

Clock f = 1 GHz --> Clock period = 1/f = $1\text{x}10^{-9}$ seconds.

Loading the program:

SSD --> DRAM = 10^6 clock cycles (first byte) + 999,999 clock cycles (subsequent bytes)

$2x10^6$ * $1x10^{-9}$ = (~$2x10^{-3}$ seconds)

(a) {

Running the Program at 90% Probability:

90% * 1,000,000 = 900,000 (number of instructions probably in cache)

10% * 1,000,000 = 100,000 (number of instructions probably in DRAM)

(1 Clock Cycle * 900,000) + (200 Clock Cycles * 100,000) = 900,000 + 20,000,000

= 20,900,000 Clock Cycles (~$20.9x10^{-3}$ seconds)

}

(b) {

Running the Program at 95% Probability:

95% * 1,000,000 = 950,000 (number of instructions probably in cache)

5% * 1,000,000 = 50,000 (number of instructions probably in DRAM)

(1 Clock Cycle * 950,000) + (200 Clock Cycles * 50,000) = 950,000 + 1,000,000

= 1,950,000 Clock Cycles (~$1.9x10^{-3}$ seconds)

}

(c) {

Theoretical Minimum occurs when P = 100%

100% * 1,000,000 = 1,000,000 (number of instructions probably in cache)

0% * 1,000,000 = 0 (number of instructions probably in DRAM)

(1 Clock Cycle * 1,000,000) + (200 Clock Cycles * 0) = 1,000,000 + 0

= 1,000,000 Clock Cycles (~$1x10^{-3}$ seconds)

Q1:

A)
01xx
10xx
xxxx
xxxx

B) 4^4^2 = 4^16
C) 5^5^2 = 5^25

Q2:

A) 2 files (e.o and windows) are updated if e.c or v.h are changed
B) 6 files (i.e. all files) are updated if w.h is changed.
C) Yes, fire Pat because Pat forgot to put v.h in the dependencies for e.o

---

Q3:
A) First version should use FPGA since we need the design quickly, and software is not an option since the software prototype was glitchy. The second version should use ASIC because expected sales volumes are large

B) Anywhere between 0 to 249. A “reg” in Verilog may or may not result in a register/flip-flop when synthesized

C) This would potentially result in the always block running forever, since it would trigger immediately after it ran. This is an undesirable condition.

Q4:

1e6 + 1e6 cycles to read from SSD to DRAM.
1e6 cycles to run.
1e6 (P*1 + (1-P)*200) cucles to load from DRAM.
This gives 3e6 + 1e6(P + (1-P)*200) cycles in all
one cycle is 1/1GHz = 1/10^9 = 10^-9 sec

A) 23.9ms if you plug in P = 0.9
B) 13.95ms if you plug in P = 0.95
C) 4ms if you plug in P = 1
