Digital System Design

by

Dr. Lesley Shannon

Email: lshannon@ensc.sfu.ca

Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc350

Simon Fraser University

Slide Set: 16

Date: April 6 2009

Slide Set Overview

- Quizzes
  - Results
  - Answers

Quizzes

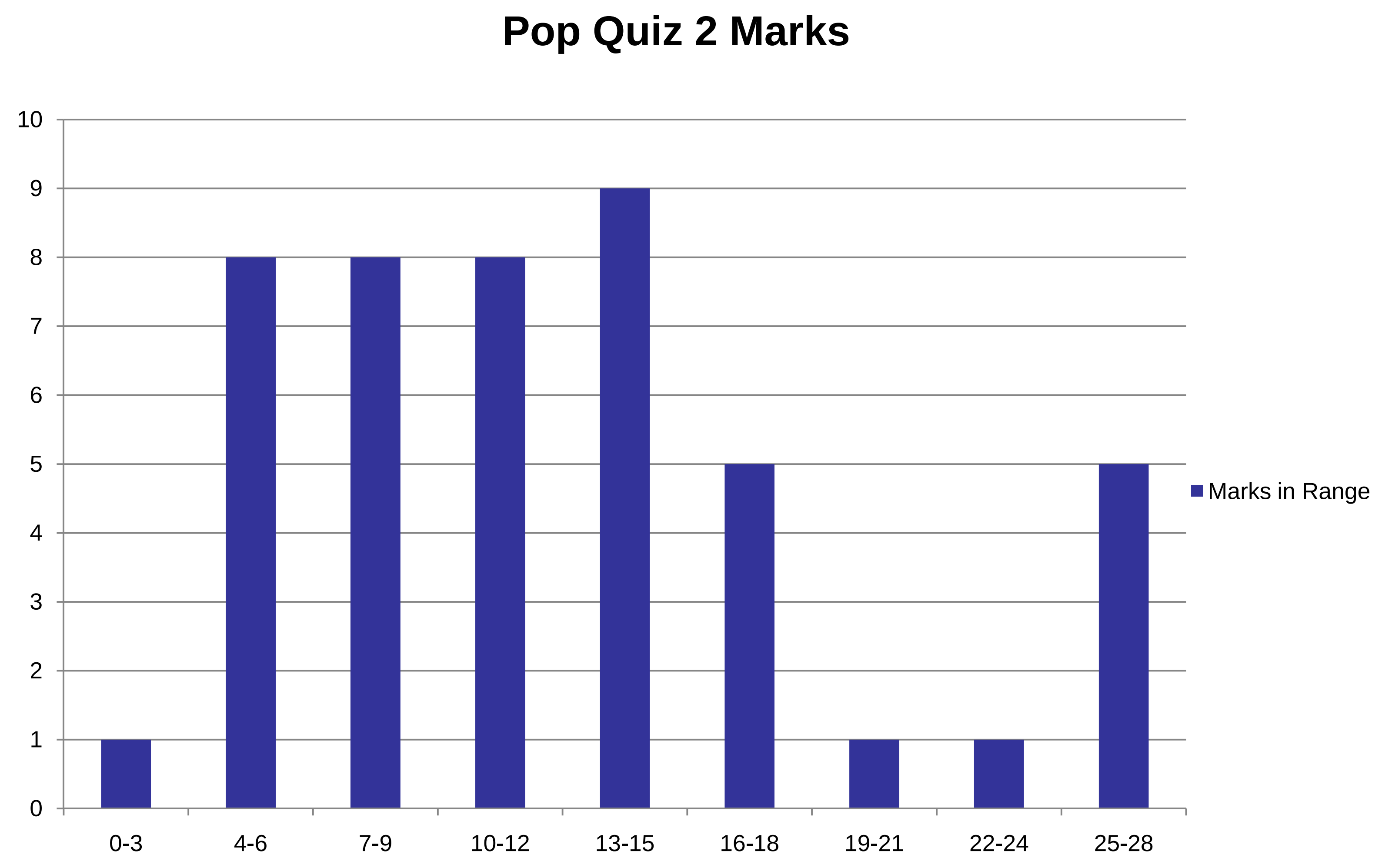
What your marks mean in terms of your final mark

• 28	7.5%
• 25	6.75%
• 22	6%
• 19.5	5.25%
• 16.5	4.5%
• 14	3.75%
• 11	3%
• 8.5	2.25%
• 5.5	1.5%

How everyone did do

- The "bad" questions
  - All of them
  - 1.(i): Avg = 6
  - 1.(ii): Avg = 6
  - Bonus: Avg = 1 (But most people didn't do the question)
- Average
  - Mean: 12.6 Median: 11.5 Mode: 8
  - Maximum: 28 Minimum: 3 STD Dev: 6.65

How everyone did do



1. (i) Rotator Register Architecture

1. (i) Rotator Register Architecture

1. (ii) Rotator Register Structural VHDL

1. (ii) Rotator Register Structural VHDL

1. (ii) Rotator Register Structural VHDL

BONUS: Rotator Register Behavioural VHDL

BONUS: Rotator Register Behavioural VHDL

Survey Stats

On average, people felt that 43% of the HDL material was new

- Number of people that felt the lecture pace was:
  - Slow: 2　　　　Fine: 23　　　　Fast: 12
- Labs were:
  - Bad: 16
  - Neutral/Mixed: 16
  - Positive: 5
- Labs took too long?
  - Should have required ~10 hrs/week for avg student (up to 12 for weak student)
    - On average: **Lab 2 – 50-60 hrs**　　　　**Lab3 – 40-50 hrs**

Survey Stats

- Common complaints about the labs were:
  - Lab 2 too long
  - Lab 2 handout should include more testbed material
    - Okay
  - Testbeds and Testbenches are hard to debug and time consuming
    - Provide more guidance on being methodical in testing
    - No one designed a high-level testbench for their system ***before*** putting it on the board (this would significantly increase your time)
  - Lab 3 handout too vague
    - I don’t understand this comment, the problem is fully defined.
- My proposed Sol’n: Incorporate design examples that ***you*** do into the class?

I greatly appreciate your feedback and look forward to your additional feedback on my evaluations.

Also, I will happily meet with anyone who wants to provide me with additional suggestions/feedback. Emails are great too.

Floating Point Operators

- Operating on Floating Point Data consists of three steps:
  - Normalization: Transfer both numbers to the same base (exponent)
  - Data operation: Do the actual calculation (add/sub/etc)
  - Denormalization: Return the result to proper floating point format based on the IEEE standard

Floating Point Operators

- Add :
  - 2.2E4 and 1.2E2
- Normalize:
  - 2.2E4 and .012E4
- Perform Addition Operation:
  - 2.212E4
- Denormalize:
  - For base ten, already done (2.212E4)

Floating Point Operators

- Add in Base two:
  - 1.111E4 and 1.11E2
- Normalize:
  - 1.111E4 and .0111E4
- Perform Addition Operation:
  - (1.111+ .0111)E4 = 10.0001E4
- Denormalize:
  - 1.00001E5
    - Store 00001 as Mantissa
    - The exponent is 5 (For storage, 132 after the 127 bias is applied)

Normalizing and Denormalizing take time approx 20% of FP calculation time

Other Floating point examples

- Other examples of stored Single Precision floating point:
  - 3F80 0000= 0011 1111 1000 0000 0000 0000 0000 0000

    = ?
  - C000 0000=1100 0000 0000 0000 0000 0000 0000 0000

    = ?
  - 0000 0000 = 0
  - 7F80 0000 = Infinity
  - FF80 0000 = - Infinity

“Recipe” for structural circuit design

- Circuit design specs are essentially “***word problems***” (recall math class in elementary school?)
- You need to determine the:
  - Inputs and Outputs
    - Bit width, types, etc
  - How to interface with the “outside”
    - i.e. additional circuitry, I/O, etc
  - What set of operations need to be performed
    - Pseudocode ***may*** be helpful

Step 1: Draw the big picture

- Include all of the input and output signals specified in the specification
  - Be sure to note bit-widths of datapaths
- Is the circuit
  - Synchronous
  - Asynchronous
  - Multi-Clock Domain
- What sequence of operations do you need to read in data
- What sequence of operations do you need to write out data
  - Be meticulous about the order and the signals affected

Step 2: What's inside the box's Datapath

- What sequence of operations do you need to perform
  - What operators do you need to design them?
- ***Draw it out***
  - The big picture (don't worry about the detailed control signals for the datapath yet)
  - Note: pseudocode *might* help
- Is the circuit multi-staged or pipelined
  - May be dictated by timing requirements
  - Also Remember: Synchronous, Asynchronous, Multi-Clock Domain

Step 3: What's inside the box's Control Path

- What state machine is needed for the control path for your data operations (be sure to include I/O and the datapath)
  - State diagrams/tables and Karnaugh maps are used here
- ***Draw it out***
  - Figure out what control signals go back to the datapath from step 2
- Is the circuit multi-staged or pipelined
  - Figure out whether multiple data words are in flight at the same time and where to put pipeline registers in your design

Step 4: Refine and finish your design

- Combine the datapath and control path
  - Make sure that your control path incorporates how your circuit reads in, processes, and writes out data correctly

- “Time step” through your design to make sure that your datapath and control path perform as you wish

- If you are writing structural HDL, label this diagram and then build your components from what you’ve drawn in the diagram

“Recipe” for behavioural circuit design

- Circuit design specs are essentially “***word problems***” (recall math class in elementary school?)
- You need to determine the:
  - Inputs and Outputs
    - Bit width, types, etc
  - How to interface with the “outside”
    - i.e. additional circuitry, I/O, etc
  - What set of operations need to be performed
    - Pseudocode ***more likely*** be helpful

Step 1: Draw the big picture

- Include all of the input and output signals specified in the specification
  - Be sure to note bit-widths of datapaths
- Is the circuit
  - Synchronous
  - Asynchronous
  - Multi-Clock Domain
- What sequence of operations do you need to read in data
- What sequence of operations do you need to write out data
  - Be meticulous about the order and the signals affected

Step 2: What's inside the box's Datapath

- What sequence of operations do you need to perform
  - What operators do you need to design them?
- ***Draw it out/Write pseudocode***
  - The big picture (don't worry about the detailed control signals for the datapath yet)
  - Note the different processes for different components
- Is the circuit multi-staged or pipelined
  - May be dictated by timing requirements
  - Also Remember: Synchronous, Asynchronous, Multi-Clock Domain

Step 3: What’s inside the box’s Control Path

- What state machine is needed for the control path for your data operations (be sure to include I/O and the datapath)
  - State diagrams/tables and Karnaugh maps are used here
- Generate the appropriate Moore/Mealy State Machines**
  - Figure out what control signals go back to the datapath from step 2
- Is the circuit multi-staged or pipelined
  - Figure out whether multiple data words are in flight at the same time and where to put pipeline registers in your design

**Drawing may still be helpful

Step 4: Refine and finish your design

- Combine the datapath and control path
  - Make sure that your control path incorporates how your circuit reads in, processes, and writes out data correctly
- Map your design to synthesizable processes
  - You may be able to do this from pseudocode, you may want a block diagram
- “Time step” through your design to make sure that your datapath and control path perform as you wish

Note: “Draw out the datapath” is the hardest part. I cannot think of a “recipe” for this. It takes practice, so you are going to have to try examples to get better at this.

There will be design question(s) on your final exam.