**National Cheng Kung University**

**Department of Electrical Engineering**

***Introduction to VLSI CAD (Spring 2018)***

**Lab Session 4**

**Decoders, Adders and the Conversion between RGB and Grayscale**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | | |
| 劉宥辰 | E24056310 | | |
| Practical Sections: | | Points | Marks |
| Prob A | | 30 |  |
| Prob B | | 30 |  |
| Prob C | | 40 |  |
| Notes | | | |

**Due Date: 15:00, April 11, 2018 @ moodle**

**Deliverables**

1. All Verilog codes including testbenches for each problem should be uploaded.

NOTE: Please **DO NOT** include source code in the paper report!

1. All homework requirements should be uploaded in this file hierarchy.
2. NOTE: Please **DO NOT** upload waveform files.

Fig.1 File hierarchy for Homework submission

**Objectives:**

**To make you be familiar with some designs of combinational logic, like adders, decoders and some operation that uses multiplication and addition. You can follow this document to practice, or you have a cleverer way. Please show your best.**

Note that you can extend the spacing if it is not enough for you to answer.

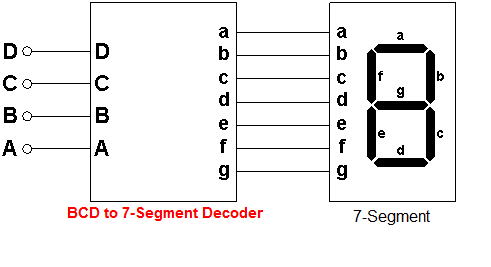
Prob A: A BCD to 7-Segment Decoder

1. **Understand the function of the target component.**

**Lab exercise 1-1: Write the Boolean function according to the truth table.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **10~15** | | | | **0** | **1** | **1** | **1** | **1** | **1** | **0** |

**Table 1.** Truth table



a=

b=

c=

d=

e=

f=

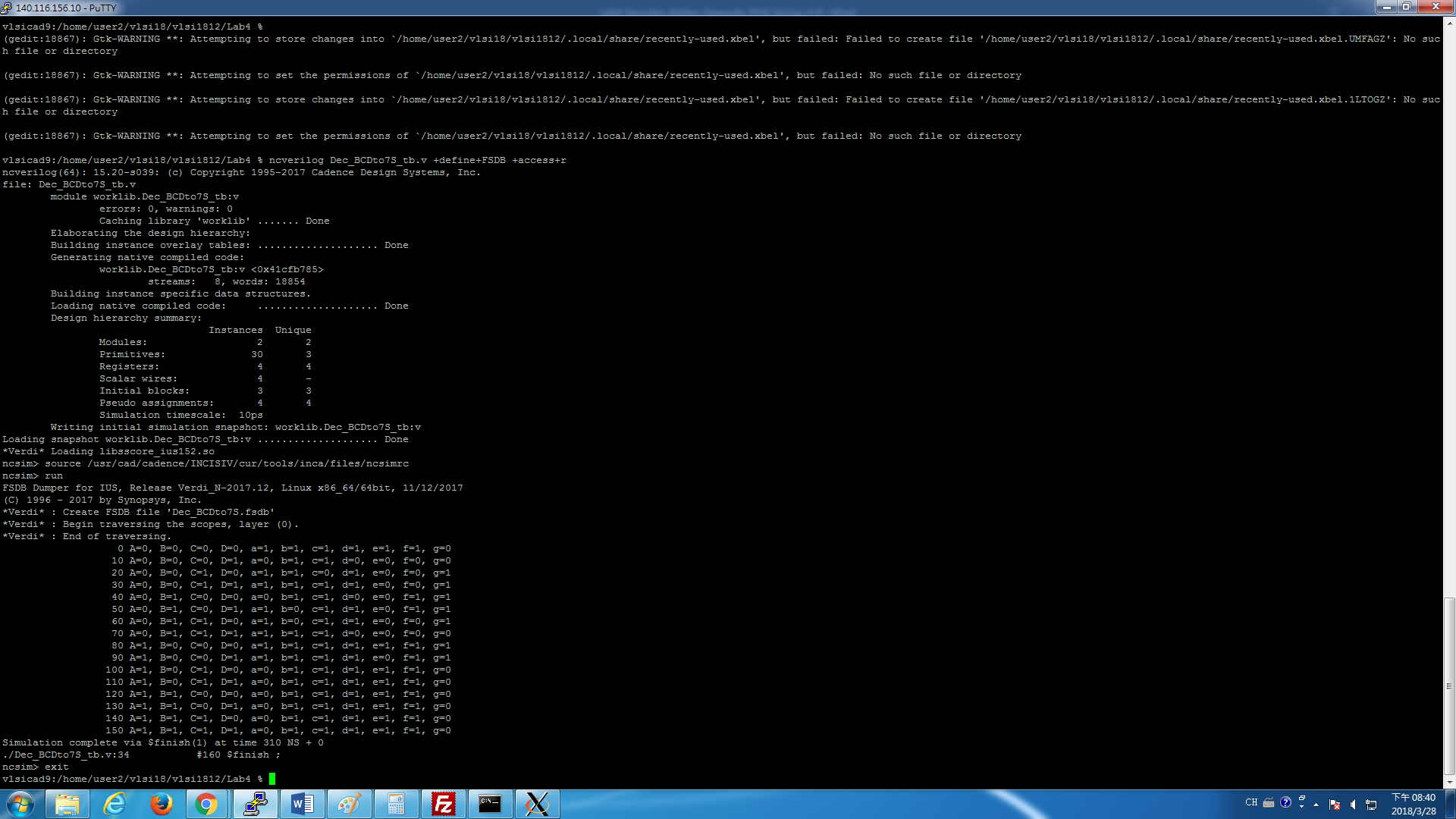
g=

1. **Design your Verilog code & testbench.**

Design the circuit according to the Boolean function at gate level. Design the testbench as well.

* 1. Name your design file *Dec\_BCDto7S.v* and your testbench file *Dec\_BCDto7S\_tb.v.*
  2. The frame code is given.
  3. Inputs: **A**, **B, C, D**
  4. Outputs: **a, b, c, d, e, f, g**
  5. Include all needed Verilog files in your testbench.
  6. Please verify the following test pattern in your testbench .

{A, B, C, D} from 0 to 15, i.e. 0000, 0001, 0010, …, 1111.

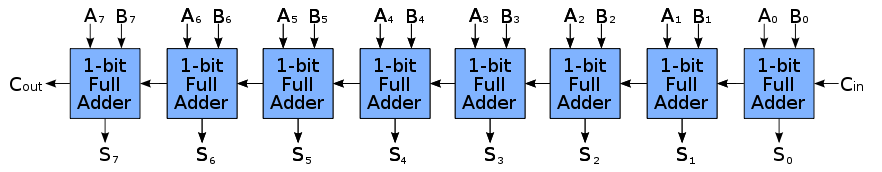


1. **Please attach your design waveforms.**

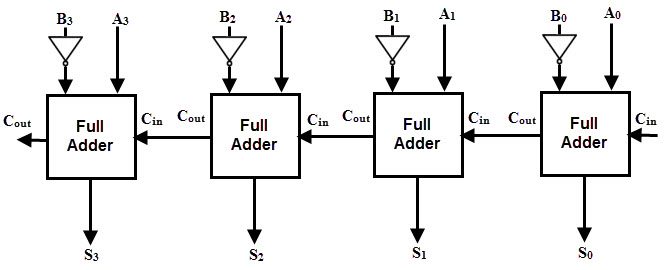
|  |
| --- |
| Your waveform : |
|  |
| Explanation of your waveform : |
| K-maps method to get the truth table.  Any then fuck the shit up. |

Prob B: An 8-bit Adder/subtractor

1. **Draw the block diagram of 8-bit adder/subtractor.**

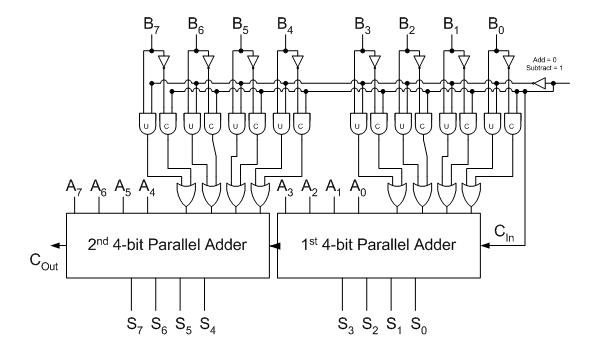
****

**Reference : https://www.electronicshub.org/binary-adder-and-subtractor/**

****

**Reference:**

**https://jawadsblog.wordpress.com/2010/03/19/an-8-bit-addersubtractor-unit/**



1. **Design your Verilog code & Testbench.**

Write an using 1-bit full adder at gate level with the following specifications. Write the testbench as well.

* 1. Number format: unsigned or 2’s complement
  2. Name your design file *addsub\_8bit.v* and your testbench file *addsub\_8it\_tb.v.*
  3. Inputs: **A[7:0]**, **B[7:0], addsub**
  4. Outputs: **S[7:0]**, **cout, ov\_flag**
  5. **A[7]**, **B[7]** and **S[7]** are the most significant bits for **A**, **B** and **S**, respectively.
  6. If **addsub = 0**, A+B will be performed, otherwise, A-B will executed.
  7. If the result causes a carry-out, **cout** will be flagged to logic 1, otherwise, 0.
  8. If the result is out of range (overflowed), **ov\_flag** will be flagged to logic 1, otherwise, 0.
  9. Don’t paste your Verilog code in the report.
  10. Include all Verilog files in your testbench
  11. Please verify the following test pattern in your testbench .
      1. **2+1, 00000010 + 00000001**
      2. **7+(-5) 00000111 + 11111010**
      3. **8+(124) 00001000 + 01111111**
      4. **25-(13), 00011001 - 00001101**
      5. **28-(-125), 00011100 - 10000011**
      6. **(-49)-(53), 11001111 - 00110101**
      7. **(-116)-(24), 10001100 - 00011000**
      8. **(-127)+(-127), 10000001 + 10000001**
      9. **128+(128), 10000000 + 10000000**

1. **Please attach your design waveforms.**

|  |
| --- |
| Your waveform : |
|  |
| Explanation of your waveform : |
| Just follow the diagram above |

Occurs problem : Memory or an array reference requires an index

Reference : <http://www.designers-guide.org/Forum/YaBB.pl?num=1483427753/6>

OK, the problem is because you're using a very old version of AMS Designer. IUS8.2 was released in late 2008/early 2009 - so the version  you're using is probably 8 years old. Why do you need to use something so old?

Prob C: Design a circuit “grayscale conversion”



1. **Design your Verilog code with the following specifications:**

This conversion unit changes 24-bit RGB values into 8-bit grayscale values. The operation is as follows:

y = 0.3125r + 0.5625g + 0.125b

0.3125 = 0.25 + 0.0625 = 1/4 + 1/16

0.5625 = 0.5 + 0.0625 = 1/2 + 1/16

0.125 = 1/8

where r, g, b are the values of segments of the 24-bit input respectively, and y is the 8-bit output.

* 1. Number format: unsigned numbers.
  2. Name your design file *grayscale.v* and your testbench file *grayscale\_tb.v.*
  3. The frame code and testbench are given. Use the testbench to help you verify the result. You don’t need to write another testbench in this problem.
  4. Inputs: **color[23:0]**
  5. Outputs: **gray[7:0]**
  6. Signal **color** represents the RGB value**.**
  7. Signal **gray** represents the grayscale value.
  8. The result of the operation should round to an integer. If you find some of your result has a slightly difference, like 1, from the expected, it is probable that the rounding is not correct.
  9. Do not use “\*” (multiply) directly in your code.
  10. Follow the PPT file to set the constrains when synthesis.
  11. Turn in the synthesized Verilog file named *grayscale\_syn.v*, the SDF file named *grayscale\_syn.sdf* and the DDC file named *grayscale\_syn.ddc*.
  12. The simulation command is different for synthesis:

ncverilog grayscale\_tb.v +define+FSDB+syn +access+r

1. **After you synthesize your design, you may have some information about the circuit. Fill in the following form.**

|  |  |  |
| --- | --- | --- |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
|  |  |  |

1. **Please attach your design waveforms.**

|  |
| --- |
| Your waveform : |
|  |
| Explanation of your waveform : |
| RGB to grayscale |

**At last, please write the lessons learned from this lab session, or some suggestions for this lab session. Thank you.**

**Simulation Command**

**ProbA: ncverilog Dec\_BCDto7S\_tb.v +define+FSDB +access+r**

**ProbB: ncverilog addsub\_8bit\_tb.v +define+FSDB +access+r**

**ProbC: RTL: ncverilog grayscale\_tb.v +define+FSDB +access+r**

**Synthesis: ncverilog grayscale\_tb.v +define+FSDB+syn +access+r**

