Verilog Lab.
Two’s Complement Add/Sub Unit

TA: Xin-Yu Shi
genius@access.ee.ntu.edu.tw
Introduction

In previous lecture, what you have learned:
- Complete representation for binary negative number
- Arithmetic operation of binary number
- Skills to design combinational circuits
- Basic concept and syntax in Verilog-HDL

In this lab, we'll guide you to model a combinational add/sub unit with gate-level Verilog HDL

SILOS: Verilog design / debug / simulation IDE
Guideline

1. Analyze specified function, divide problems into sub-problems and conquer them one by one

2. Figure the architecture

3. Coding

4. Simulation
Block Diagram

4-bit 2's Complement Add/Sub

<table>
<thead>
<tr>
<th>mode</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>add</td>
</tr>
<tr>
<td>1</td>
<td>subtract</td>
</tr>
</tbody>
</table>
Input/Output Description

Input signals
- \texttt{operand\_a, operand\_b}: signed 4-bit integer
- \texttt{mode}: select the operation
  - 0: result = operand\_a + operand\_b
  - 1: result = operand\_a – operand\_b

Output signals
- \texttt{result}: signed 4-bit integer
Two’s Complement (1/2)

\[ [N]_2 = 2^n - (N)_2 \]

- **EX:** 2’s complement of \((N)_2 = (01100101)_2\)

  \[ [N]_2 = [01100101]_2 \]
  \[ = 2^8 - (01100101)_2 \]
  \[ = (100000000)_2 - (01100101)_2 \]
  \[ = (10011011)_2 \]

- **EX:** show that \((N)_2 + [N]_2 = 0\)

  \[
  \begin{array}{c}
  01100101 \\
  + 10011011 \\
  \hline
  101011111 \\
  \quad (000000000)
  \end{array}
  \]
  \[
  [N]_2 = - (N)_2
  \]
Two’s Complement (2/2)

Convert \((N)_2\) to \([N]_2\):

\[
a_k \rightarrow \bar{a}_k, \quad \begin{cases} 1 \rightarrow 0 \\ 0 \rightarrow 1 \end{cases} \quad \text{then add 1}
\]

\[
N = 01100101 \\
10011010 \quad \text{complement each bit}
\]

\[
\begin{array}{c}
10011010 \\
\hline
110011011
\end{array}
\quad \text{add 1}
\]
Unsigned Adder

- Unsigned addition by carry ripple adder (CRA)
- Implement with 1-bit full adders
- For N-bit addition, each bit:
  - \( \text{Sum}_i = 1 \), if the number of 1 in \( \{a_i, b_i, \text{carry}_{i-1}\} \) is odd
  - \( \text{Carry}_i = 1 \), if the number of 1 in \( \{a_i, b_i, \text{carry}_{i-1}\} \) is equal to or more than 2
Implement Add/Sub with Unsigned Adder

- Divide and conquer
  - **add**
    - signal connection
    - module instance
  - **subtract**
    - Identical:
      - signal connection
      - module instance
Architecture Design

- Share the 4-bit carry ripple adder
- For subtract, inverse $b_3 \sim b_0$ and add 1 from carry in

\[
\begin{array}{c|c|c|c|c|c|c|c|}
\text{mode} & b_i & \text{operand}_b_i \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

\[
\text{operand}_b_i = \text{mode} \oplus b_i
\]
Lab Procedures

Please follow the steps to build the dual-mode adder hierarchically.

1. Download the Verilog source code files from: http://access.ee.ntu.edu.tw/course/logic_design_95first/VerilogLabSource.zip

2. Complete/modify the source code

3. Simulate by SILOS
1. Full Adder Gate-Level Model

Please complete the Verilog code of 1-bit full adder. *(FA_1bit.v)*

```verilog
module FA_1bit( sum, co, a, b, ci);

  input  a, b, ci;
  output sum, co;

  Fill in the necessary code
  (Please only use AND/OR gate)

endmodule
```
2. Carry Ripple Adder Model

Debug the Verilog code of 4-bit carry ripple adder.

(CRA_4bit.v)

```verilog
module CRA_4bit( sum, co, a, b );

input [3:0] a, b;
input ci;
output [3:0] sum;
output co;

reg [2:0] carry_out;

FA_1bit fa0 ( carry_out[0], sum[0], a[0], b[0], ci );
FA_1bit fa1 ( carry_out[1], sum[1], a[1], b[1], carry_out[0] );
FA_1bit fa2 ( carry_out[2], sum[2], a[2], b[2], carry_out[1] );
FA_1bit fa3 ( co , sum[3], a[3], b[3], carry_out[2] );

endmodule
```
3. Add/Sub Unit Model

Complete the Verilog code of the add/sub unit (Add_Sub_Unit.v)

```verilog
module Add_Sub_Unit( result, operand_a, operand_b, mode );

input [3:0] operand_a, operand_b;
input mode;
output [3:0] result;

wire [3:0] xor_b;

xor g0
xor g1
xor g2
xor g3

CRA_4bit m1( .sum , .co , .a , .b , .ci );
endmodule
```

Fill in the necessary code
4. Testbench

Please complete the module instance in testbench.

(tb_Add_Sub_Unit.v)

```verbatim
module tb_Add_Sub_Unit;

    // for DUT port connection
    reg     mode;
    reg [3:0] operand_a, operand_b;
    wire [3:0] result;

    // for pattern generation
    integer    a, b, expect, error_num, t;
    reg        overflow;

    // instance the design under test here
    Fill in the necessary code

    // give stimulus pattern
    initial begin
        error_num = 0;

        $display("\nTest addition mode.");
        mode = 1'b0;
```
Simulation using SILOS
Create new project

- Write your Verilog design code
- Create a new project:
  - Project -> New -> (enter your project name)
- Include your verilog file(s):
  - (choose your .v file) -> (click Add bottom) -> (click OK)
  - Note that testbench should be included
- Then run and debug
Type project name

1. Select Verilog source file/testbench file
2. Add to project
3. Click OK
Run and Debug

Run

- Press F5 or click the “GO” icon on toolbar
- An output window is generated to show whether there exists errors.

Debug

- Press F6 then click the “Open explorer” icon on toolbar
  - Choose signals you want to watch
  - Click right button and select “Add signal to analyzer”
- Then you can debug with waveform
Press Go (F5)

Note if there is any warning or error

Highest level modules (that have been auto-instantiated):
(testbench_FA testbench_FA
12 total devices.
Linking ...

13 nets total: 21 saved and 0 monitored.
69 registers total: 69 saved.
Done.

0 State changes on observable nets.

Simulation stopped at the end of time 0.
Ready: sim

78 State changes on observable nets.

Simulation stopped at the end of time 100.
Ready:
1. Drag the signal you want to observe

2. Drop here

3. Observe the waveform
Questions (1/2)

Please write Verilog codes for dual-mode 8-bit Add/Sub circuit design. (Similar to the architecture on p.10)

Download the needed Verilog codes (including test-bench & template files) from:

http://access.ee.ntu.edu.tw/course/logic_design_95first/VerilogLabQuestion.zip

Please write your whole design in only one Verilog file. (You can concatenate all the modules in only one file.)

Deadline: E-mail to genius@access.ee.ntu.edu.tw before 2:00 PM 12/08.
Questions (2/2)

Naming rule: For the student with student number of B94901101, the hand-in file should be B94901101.v; otherwise, the grade will be empty.

Template Code for Top Module:

```verilog
module Add_Sub_Unit(
    result, operand_a, operand_b, mode);

parameter word_length = 8;

input [(word_length-1):0] operand_a, operand_b;
input mode;
output [(word_length-1):0] result;

endmodule
```