Verilog Lab.
2’s Complement Add/Sub Unit

TA: Chihhao Chao
chihhao@access.ee.ntu.edu.tw

Lecture note ver.1 by Chih-hao Chao
Introduction

In previous lecture, you learned
- Complete representation for binary negative number
- Arithmetic operation of binary number
- Skills to design combinational circuits
- Basics of Verilog HDL

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

Experience SILOS, a 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

<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
- operand_a, operand_b: signed 4-bit integer
- mode: select the operation
  - 0: result = operand_a + operand_b
  - 1: result = operand_a – operand_b

Output signals
- result: signed 4-bit integer
2’s Complement

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

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 \((N)_2 + [N]_2 = 0\)

\[
\begin{array}{c}
01100101 \\
+ 10011011 \\
\hline
00000000
\end{array}
\]

(carry)

\[ [N]_2 = - (N)_2 \]
2’s Complement

Convert \((N)_2\) to \([N]_2\): way 2

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

\[
\begin{array}{cccccccc}
N &=& 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & & \rightarrow \text{ complement each bit} \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & & \rightarrow \text{ add 1} \\
\end{array}
\]
Unsigned Adder

- Unsigned addition by carry ripple adder
- 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

\[
\begin{align*}
1111 & \quad \text{進位} \\
13_{10} & = 1101 \\
11_{10} & = 1011 \\
11000 & = 24_{10}
\end{align*}
\]
Implement Add/Sub with Unsigned Adder

Divide and conquer

- **add**

  ![Adder Diagram](image)

  - Identical:
    - signal connection
    - module instance

- **subtract**

  ![Subtractor Diagram](image)
Architecture Design

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

<table>
<thead>
<tr>
<th>mode</th>
<th>$b_i$</th>
<th>operand_b_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$\text{operand}_b_i = \text{mode} \oplus b_i$
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_94first/

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.

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

input a, b, ci;
output sum, co;
wire ab, bc, ca;

dmodule

C.H. Chao, 11/18/2005
2. Carry Ripple Adder Model

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

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

input [3:0] a, b;
inout 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

Please complete the Verilog code of the add/sub unit

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
4. Testbench

Please complete the module instance in testbench

```verilog
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

  // give stimulus pattern
  initial begin
    error_num = 0;
    $display("\nTest addition mode." );
    mode = 1'b0;
    for( a=-8; a<=7; a=a+1 ) begin
      for( b=-8; b<=7; b=b+1 ) begin
        operand_a = a[3:0];
        operand_b = b[3:0];
        expect = a + b;
```
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
1. Select Verilog source file/testbench file
2. Add to project
3. Click OK

Type project name
Run and Debug

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

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

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:

Note if there is any warning or error
1. Drag the signal you want to observe

2. Drop here

3. Observe the waveform
Questions

1. What happens when you click GO button? What message did you see on screen?

2. Please add 5+ (-7). What is the answer? What is the carry out?

3. Please add 5+7, what happens? Why the sum doesn’t equal to 12? Add one output port to the add/sub unit and derive the detection logic. Redraw the architecture of add/sub unit and modify the code template in next page.

Submit with HW4
module Add_Sub_Unit( result, operand_a, operand_b, mode, detect );

input [3:0] operand_a, operand_b;
input mode;
output [3:0] result;
output detect; // for question 3

wire [3:0] xor_b;

xor g0 ( xor_b[0], operand_b[0], mode );
xor g1 ( xor_b[1], operand_b[1], mode );
xor g2 ( xor_b[2], operand_b[2], mode );
xor g3 ( xor_b[3], operand_b[3], mode );

CRA_4bit ml(
  .sum
  .co
  .a
  .b
  .ci
);

// add your code for detection
// declare net(wire) you need

// connect gate with net

endmodule