NYCU-EE IC LAB - Spring 2023

Lab06 Practice

Design: Complex Multiplier (Genvar Version)

Objective

This practice will guide you to use different **genvar** statement in different kind of scenario.

Data Preparation

1. Extract test data from TA's directory:

   % tar xvf ~iclabta01/Lab06.tar

2. The extracted LAB directory contains:

   a. **00_TESTBED**

   b. **01_RTL**

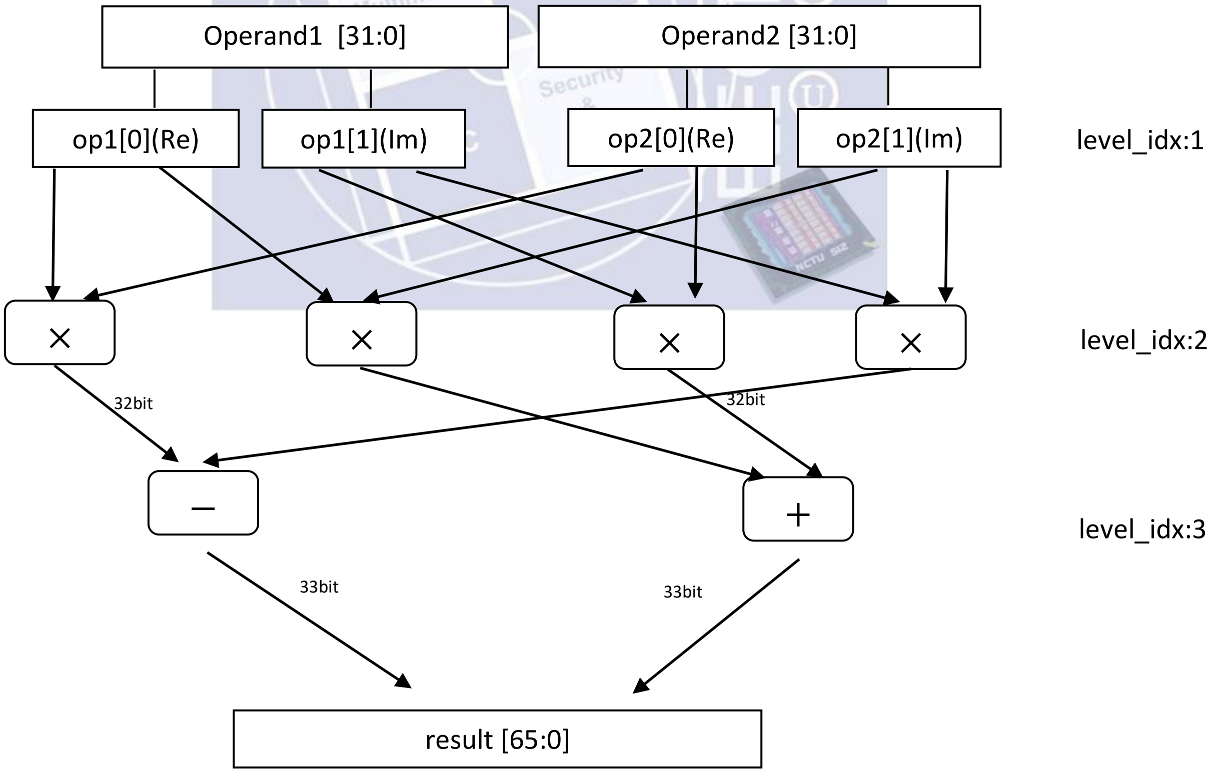
   c. **02_SYN**

   d. **03_GATE**

Design Description

Design a complex number multiplier with genvar method as practice.

Circuit Architecture: **GENVAR_PRAC.v**



Inputs and Outputs

Input signal	Bit width	Definition
operand1	32	Input operand1 {real[15:0],imagine[15:0]}
operand2	32	Input operand2 {real[15:0],imagine[15:0]}

Output signal	Bit width	Definition
result	66	Output number {real[32:0],imagine[32:0]}

COMPLEX_MULT.v : Just wrapper to demonstrate for using your own IP

GENVAR_PRAC.v : To practice genvar statement

Syntax

```
for (genvar_name = constant_expression; constant_expression; genvar_name = constant_expression)
begin: generate_block_name
        generate_item
        generate_item
        …
        if (constant_expression) begin: generate_block_name
                generate_item
                for (….. )begin: generate_block_name
                   generate_item
                end
        end
end
```

Summary

1. wire or reg: could be instantiated in the **for loop** or **if loop**
2. if-else condition could only apply for genvar variable
3. each iteration has only the scope of that iteration. To access previous iteration's item , you must specify the item hierarchically by using generate_block_name.
4. Strongly Recommend using nWave to see the architecture of your genvar statement design.

Version

2018 Fall: Li-Wei Liu