Computer-Aided VLSI System Design, Fall 2015

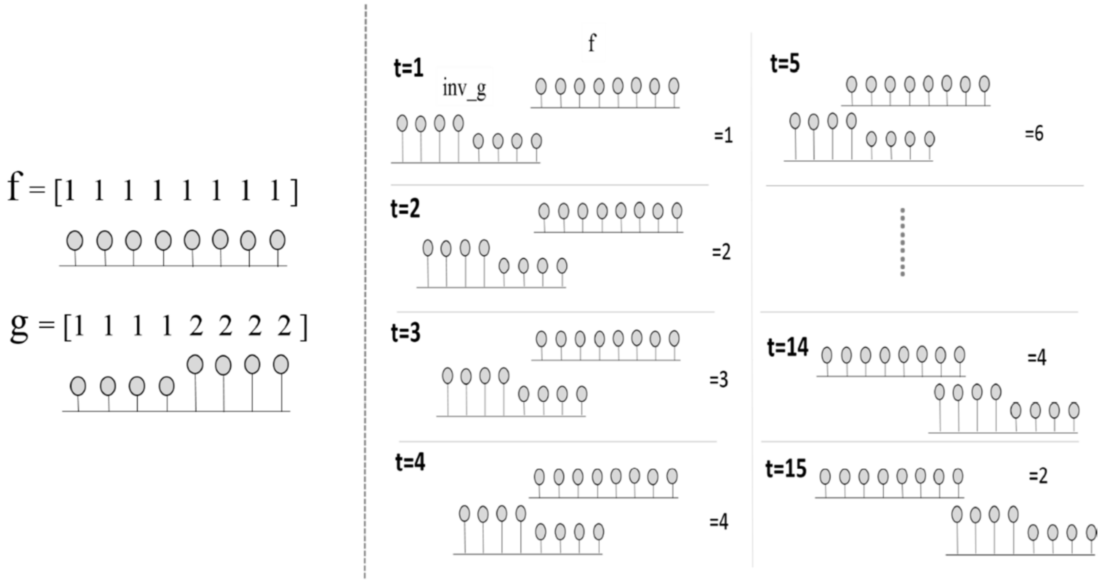
Verilog Homework 2

1. Function Description

Convolution is the most important and fundamental concept in signal processing and analysis. By using convolution, we can construct the output of a system for any arbitrary input signal, is we know the impulse response of the system. In this homework, we will implement a discrete convolution calculator to compute the convolution of two signal, f and g. The discrete convolution is given by:

$$(\boldsymbol{f} * \boldsymbol{g}) \overset{\text{def}}{=} \sum_{\boldsymbol{m}=-\infty}^{\infty} \boldsymbol{f}[\boldsymbol{m}]\boldsymbol{g}[\boldsymbol{n}-\boldsymbol{m}]$$

In this design, we are going to implement 8-point-by-8-point discrete convolution. The example is shown below:



$$\boldsymbol{f} * \boldsymbol{g} = [1\ \ 2\ \ 3\ \ 4\ \ 6\ \ 8\ \ 10\ \ 12\ \ 11\ \ 10\ \ 9\ \ 8\ \ 6\ \ 4\ \ 2]$$

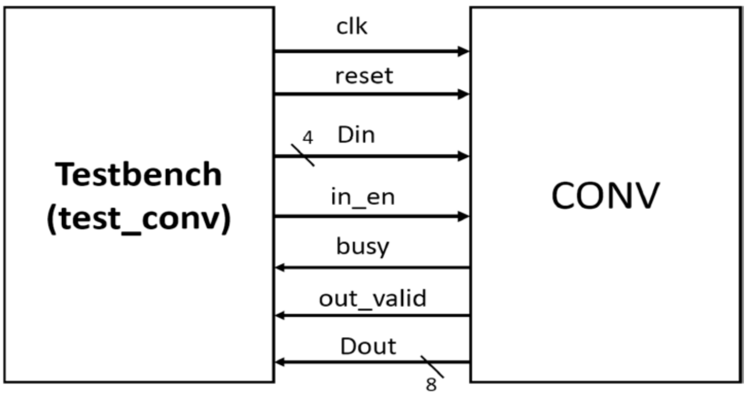


Fig. 1. I/O port connecting

2. Specification

The input/output pins are defined in the following:

Signal Name	Input/Output	Bit Width	Description
clk	Input	1	Clock signal. Positive edge trigger.
reset	Input	1	Active high asynchronous reset signal.
Din	Input	4	Input data **(one point per cycle).** Every point only transmitted once.

in_en	Input	1	Input data enable signal:
			in_en=0(disable)
			in_en=1(enable)
			If signal busy=1, the testbench would not transmit input data.

busy	Output	1	The busy signal on CONV: busy=0 (Testbench can input data). Busy=1 (Testbench cannot input data).

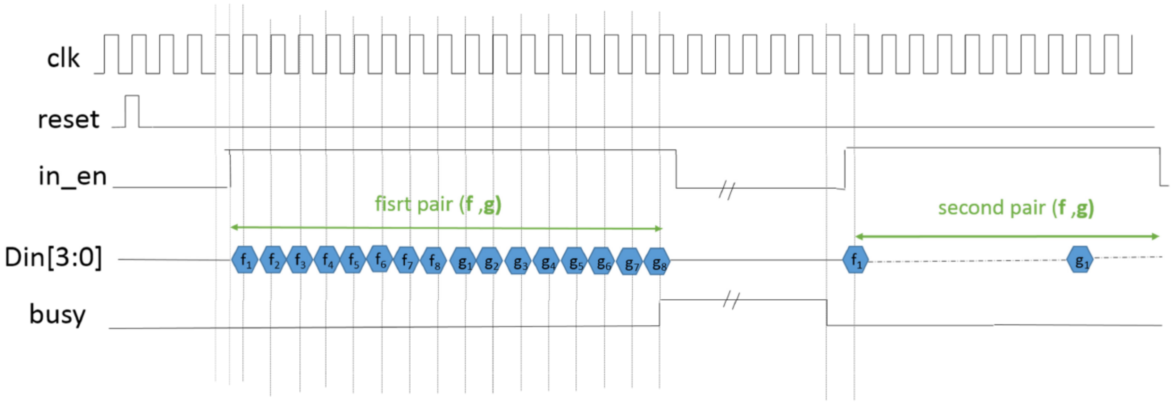
out_valid	Output	1	Ouput control signal:
			out_valid = 0 (invalid output data).
			out_valid = 1 (valid output data).

Dout	Output	8	Output data (**one point per cycle**). Don't consider overflow.

3. Timing Diagram

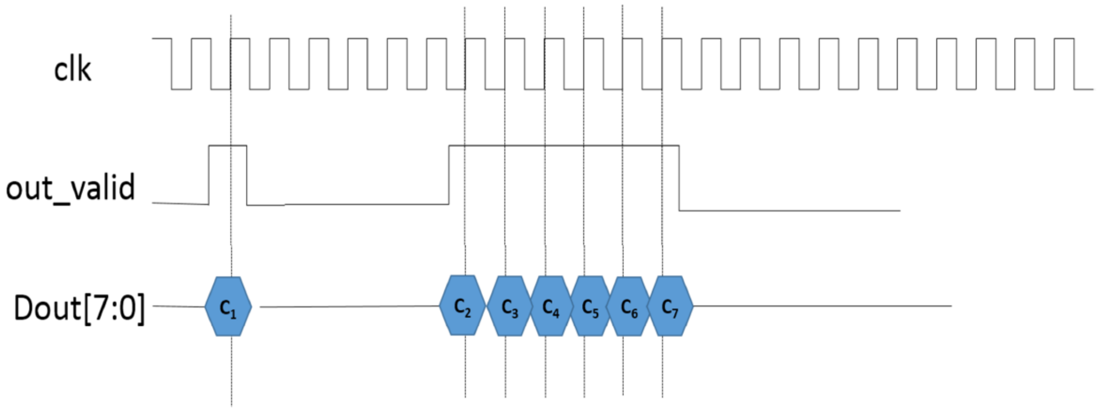
The following figure shows the input timing diagram of the convolution circuit. After the signal **in_enable** is asserted (1'b1), point values are given serially for 16 cycle. The first 8-point data are signal **f**, the next 8-point data are signal **g**, and so on.

a. After reset and wait 2 cycles
if busy==0 -> in_en = 1



b. if busy==1 -> in_en = 1

The following figure shows the output timing diagram of the convolution design. Whenever the signal **out_valid** is asserted (1'b1), the signal **Dout** will be fetched by testbench. **Note that the convolution result should be output serially as indicated in the figure.**



4. Provided Files

File Name	Description
CONV.v	Design file with I/O ports.
testfixture1.v	Testbench of CONV.v
pattern_1.dat	Test patterns of Din. All patterns are **positive**.
golden_1.dat	Answer of convolution.

5. RTL Simulation

You need to finish Verilog RTL and check its functional correctness. Here are the instructions you need to run: (Correct codes will see **Congratulation!**)

```
ncverilog +access+r testfixture1.v CONV.v
```

6. Grading Policy

Convolution of (Rectangular, Rectangular) are correct! (30%)

Convolution of (Rectangular, Upper Wave) are correct! (30%)

Congratulations! All data have been generated successfully! (30%)

Report for figures of simulation results and some conclusions. (10%)

7. Submission Requirements

Please submit your design as follows in one .zip file with the naming convention: StedentID_HW2_vk.zip (k is the number of version, k = 1, 2, …)

File Name	Description
CONV.v	Your design.

testfixture1.v	Testbench of CONV.v
pattern_1.dat	Test patterns of Din. All patterns are **positive**.
golden_1.dat	Answer of convolution.
report.pdf	Your report.

8. Submission Deadline
   2015/10/20 22:00

Appendix: Test patterns

1. Convolution of (Rectangular, Rectangular)
   [2, 2, 2, 2, 2, 2, 2, 2] * [2, 2, 2, 2, 2, 2, 2, 2]
   = [4, 8, 12, 16, 20, 24, 28, 32, 28, 24, 20, 16, 12, 8, 4]

2. Convolution of (Rectangular, Upper Wave)
   [1, 1, 1, 1, 1, 1, 1, 1] * [1, 2, 3, 4, 5, 6, 7, 8]
   = [1, 3, 6, 10, 15, 21, 28, 36, 35, 33, 30, 26, 21, 15, 8]

3. [1, 2, 3, 4, 5, 6, 7, 8] * [1, 2, 3, 4, 5, 6, 7, 8]
   = [1, 4, 10, 20, 35, 56, 84, 120, 147, 164, 170, 164, 145, 112, 64]

4. [1, 3, 5, 7, 7, 5, 3, 1] * [8, 6, 4, 2, 2, 4, 6, 8]
   = [8, 30, 62, 100, 126, 130, 124, 120, 124, 130, 126, 100, 62, 30, 8]

5. [1, 5, 2, 7, 3, 6, 1, 8] * [2, 4, 7, 1, 2, 5, 3, 6]
   =[2, 14, 31, 58, 55, 90, 86, 110, 122, 117, 91, 57, 79, 30, 48]

Appendix: Tool list

NCverilog: source /usr/cadence/cshrc