NCTU-EE ICLAB – Spring 2018

Final Project

Design: Gradient Descent

Data Preparation

1. Extract test data from TA's directory:
   % tar xvf ~iclabta01/GD.tar

Design Description

For your final project, you have to implement *Gradient Descent* on a polynomial function that is convex. Gradient descent is a 1st iterative optimization algorithm for finding the minimum of a function. Convergence is guaranteed if the function that you are optimizing on is convex. In this project a 2nd order polynomial is given as:

$$f(x) = Dx^2 + Bx + C \tag{1}$$

where $D, B, C \in \mathbb{R}$. Gradient descent is an iterative algorithm that has the following form:

$$\hat{x} = x - \alpha \frac{df}{dx} \tag{2}$$

where $\hat{x}$ is the updated value of x given an update of $\alpha \frac{df}{dx}$ ($\alpha \in \mathbb{R}^+$). The derivative of $f(x)$ with respect to x is given as:

$$\frac{df}{dx} = 2Dx + B \tag{3}$$

To keep the notations simple for our project, we take the denominator away on the first term resulting in:

$$\frac{df}{dx} = Ax + B \tag{4}$$

where $A = 2D$.

An illustration of how gradient descent work is shown below:

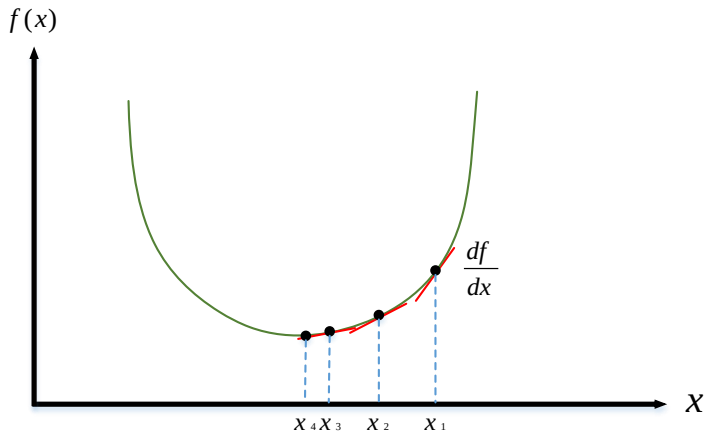


Figure 1: Gradient Descent Example

As we can see from Figure 1, it takes 3 iterations for us to find the value of x that corresponds to the minimum of $f(x)$ where the update of x is as given in Equation 2.

In this project, you'll be working with 2's complement signed values that requires fixed-point representation. An example of fixed-point representation is shown below:

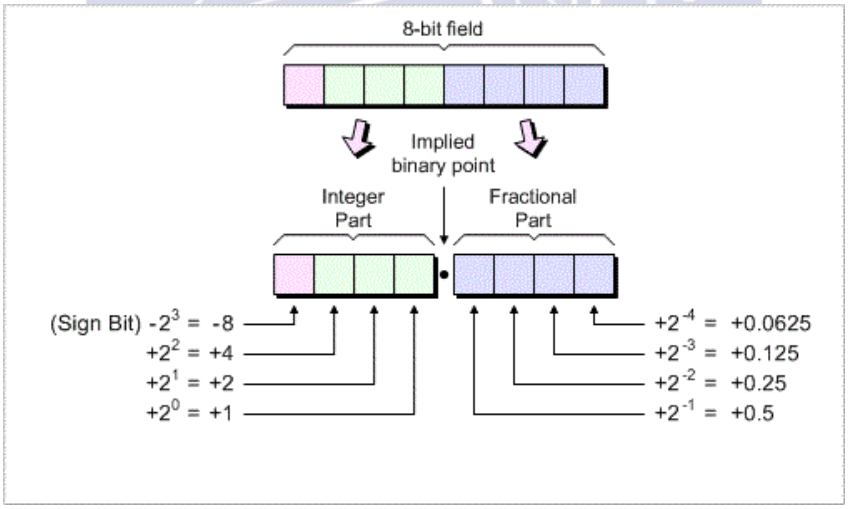


Figure 2: Fixed-point representation

This project is designed in such a way where you have to balance between the *number of iterations*, *total error* and *area*. (The *total error* is the cumulative error from the value of x you found from the exact solution of x).

Hints

You have to design your own pattern to test your design.

Note that from Equation 4, we have the following properties to keep the objective function in Equation 1 convex:

- $A \in \mathbb{Z}^+$
- $B \in \mathbb{Z}$

Two files are prepared for you to test your design:

- input.txt

```
1,-5
2,-8
3,-5
7,1
3,7
```

Format: A,B

- output.txt

```
5.000000010214032
3.999999995004006
1.6666666700343233
-0.1428571414277461
-2.3333333300026764
```

Inputs and Output

Input signal	Bit width	Definition	Note
clk	1	Clock	
rst_n	1	Asynchronous active-low reset	
in_valid	1	Enable input signal	
in_A	4	Parameter A of Equation 4	Signed. >0
in_B	4	Parameter B of Equation 4	Signed.

Output signal	Bit width	Definition	Note
out_valid	1	Enable output check	
out_data	21	Value of x corresponding to the minimum value for $f(x)$ in Equation 1	Signed. Fixed point. Integer Part: 5 bits Fractional Part: 16 bits

Specifications

1. Top module name : **GD** (File name : **GD.v**)
2. **Do not use closed-form approach to get the results in hardware implementation. Doing so will result in 0 score for this project.**
3. Upon convergence, the error ($|x - actual\ \ value|$) of should be **below 0.05** to be considered correct.
4. It is **asynchronous** reset and **active-low** architecture.
5. The reset signal would be given only once at the beginning of simulation. All output signals should be **zero** after the reset signal is asserted.
6. The max clock period of the design is **5ns** for gate-level and **10ns** for Post-Sim. You can optimize the clock cycle.
7. The in_data will be given when in_valid pulls up.
8. Once you have to optimum value for x, *out_valid* will pull up and the pattern will check the value of the output pins *out_data*.
9. The input delay is set to **0.5*clock cycle**.
10. The output delay is set to **0.5*clock cycle**, and the output loading is set to **0.05**.
11. The synthesis result of data type **cannot** include any latches.
12. After synthesis, you can check **GD**.area and **GD**.timing. The area report is valid when the slack in the end of timing report should be **non-negative**.
13. **Area** of your design must **lower than 50000**.
14. Your design should output its result within **4000 cycles**.
15. The gate level simulation **cannot** include any timing violations **without** the **notimingcheck** command.
16. The input delay of **clk** and **rst_n** should be **zero**.
17. Using **top wire load mode** and **compile ultra**.

//---------- APR ----------//

1. **If you have wrong modification of this part, layout part will be treated as failed.**

2. Timing Constraint

   You have to modify the values related to clock period and following timing specifications. TA will check this file.

3. Core power pad and Pad power pad

   (a) At least one pair at each side.

4. Floor planning

   (a) Core Size:

   Define by user, but it matters your performance

(b) Core to IO boundary :

Each side must be more than 100

5. Power planning

(a) Core Ring

Top & Bottom: metal layer must be odd and width is 9.

Left & Right: metal layer must be even and width is 9.

Each side must be wire group, interleaving, and at least 4 pairs.

(b) Stripes

Vertical: metal layer must be even and width is at least 2.

Horizontal: metal layer must be odd and width is at least 2.

Both two directions must be at least 3 pairs.

6. Timing analysis results

(a) Timing Slack:

NO negative slacks after setup/hold time analysis (include SI).

(b) Design Rule Violation (DRV)

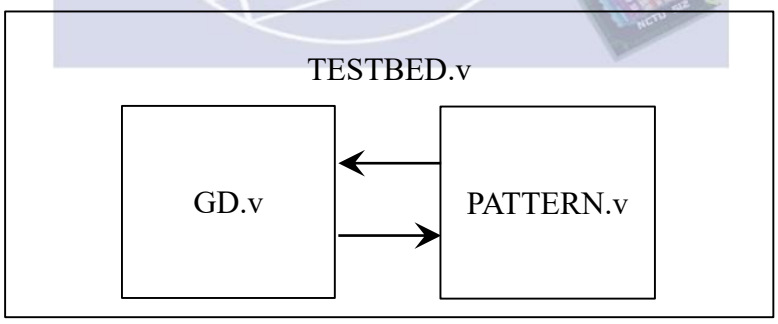
The DRV of (fanout, cap, tran) should be all 0 after setup/hold time analysis (include SI).

7. Design verification results

(a) Layout vs. Schematic (LVS) NO LVS violations after "verify connectivity".

(b) Design Rule Check (DRC) NO DRC violations after "verify geometry".

Block Diagram



Grading Policy

1. **Grading policy:**
   - Gate level simulation: 10%
   - APR correctness: 10%

- Post simulation: 10%
- Performance:
  - Accuracy: 40%
  - Total simulation time: 20%
  - Gate level Area: 5%
  - APR area: 5%

2. **Please upload the following file on e3 platform before noon (12:00 p.m.) on June. 25:**

   - GD_iclabXX.v (XX = Your account)、

     {cycle_delay_gate}_{cycle_delay_post}.txt

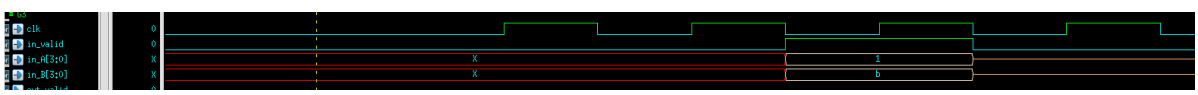
     - Ex. GD_iclab99.v、GD_iclab99_5_9.txt

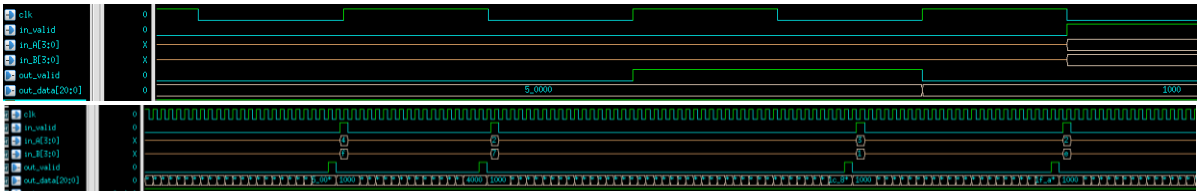
   ➢ **APR**
   - ***CHIP_ iclabXX.v***
   - ***CHIP_ iclabXX.sdf***
   - ***CHIP_ iclabXX.io***
   - ***CHIP_ iclabXX.sdc***
   - ***iclabXX.inn.dat.tar***
     - ***(unix% tar cvf iclabXX.inn.dat.tar iclabXX.inn.dat/)***
   - ***iclabXX.inn***

3. **Template folders and reference commands:**

   01_RTL/ (RTL simulation) **./01_run**
   02_SYN/ (Synthesis) **./01_run_dc**
   (Check the design if there's latch or not in ***syn.log***)
   (Check the design's timing in /Report/**GD*.timing***)
   03_GATE / (Gate-level simulation) **./01_run**
   04_MEM/ (Memory files) (no need to modify)
   05_APR/ (back-end APR) **./01_combine**
   ./02_run_uniquify
   ./09_clean_up (clean all log and command files)
   06_POST_SIM/(Post-layout simulation) **./01_run**

Example Waveform




clk
in_valid
in_A[3:0]
in_B[3:0]
out_valid
out_data[20:0]
5_0000
1000


System Integration
&
Silicon
Implementation
Group.
ELECTRONICS
NCTU
Multimedia
Wireless
FEC
Security
&
PLL